Universidade de Aveiro Departamento de Engenharia Mecânica
2019

Tiago da Silva Rufino

Sistema front end de máquina de ensaios baseado em solução Raspberry Pi

Universidade de Aveiro Departamento de Engenharia Mecânica
2019

Tiago da Silva Rufino

Sistema front end de máquina de ensaios baseado em solução Raspberry Pi

Projeto apresentado à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestrado em Engenharia Mecânica, realizada sob orientação científica de Rui António da Silva Moreira, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro e de José Paulo Oliveira Santos, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade do Aveiro.

O júri / The jury

Presidente / President	**Prof. Doutor Miguel Armando Riem de Oliveira** Professor Auxiliar da Universidade de Aveiro
Vogais / Committee	**Prof. Doutor Pedro Nicolau Faria da Fonseca** Professor Auxiliar da Universidade de Aveiro
	Prof. Doutor Rui António da Silva Moreira Professor Auxiliar da Universidade de Aveiro (orientador)

Agradecimentos / Acknowledgements

Agradeço ao Professor Rui Moreira e ao Professor José Paulo Santos por toda a ajuda e compreensão, a toda a minha família e a todos os colegas do laboratório pela inter-ajuda demonstrada

Palavras-chave Máquina de Ensaios, Sistema front-end, Interface gráfica, Raspberry Pi 3, Base de Dados, Aplicação Web.

Resumo O projecto consiste no desenvolvimento de um sistema de front end de uma maquina de ensaios do laboratório de dinâmica das estruturas do departamento de engenharia mecânica da Universidade de Aveiro. A máquina de ensaios é composta por uma atuador elétrico linear e tem como sensores um encoder de alta resolução que deteta a posição e um transdutor de força analógico que deteta a carga aplicada. A atuação e leitura de resultados é feita recorrendo ao "single-board computer" Raspberry Pi 3. O Raspberry Pi faz também a gestão da base de dados e atua como server para a conexão remota. Com o apoio dos projetos:UID/EMS/00481/2013-FCT eCENTRO-01-0145-FEDER-022083.

Keywords Testing Machine, Front-end system, Graphic Interface, Raspberry Pi 3, Database, Web Application

Abstract This projet consists in the development of a front-end system of a testing machine from the lab of structures dinamics of the University of Aveiro. The testing machine has a linear eletric actuator and has as sensors an high resolution encoder to detect the position and a analogic tranductor of force to detect the load applied. The sensoring and actuation e done by the single-board computer Raspberry Pi 3. The Raspberry Pi also manage the database used in the projet and act as server for the renote conections. With the support of the projects:UID/EMS/00481/2013-FCT eCENTRO-01-0145-FEDER-022083

Índice

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

1.1 Enquadramento

Em 2009 foi desenvolvido no departamento de Engenharia Mecânica da Universidade de Aveiro uma máquina de ensaios de materiais com o intuito de estudar o comportamento mecânico de reforços para tecidos ligamentares com fibras de compósitos biodegradáveis de base polimérica. Estes reforços iriam auxiliar na recuperação de lesões nos ligamentos pois suportariam inicialmente a totalidade da carga e seriam posteriormente absorvidos pelo organismo com o uso continuado coincidindo com a reabilitação do ligamento.

1.2 Motivações e Objetivos

O objetivo deste projeto foi criar um sistema front-end que permitisse controlar e programar a máquina de ensaios. Ao nível do controlador pretendia-se a criação de macros para poder executar 6 tipos diferentes de ensaios conjugados com diferentes geometrias do material a testar assim como o controlo manual da máquina de ensaios e outro tipo de configurações. Após o fim do ensaio, os resultados deveriam ser guardados numa base de dados para posterior consulta. Todo este controlo devia ser feito pelo computador Raspberry Pi com uma aplicação local a correr permanentemente nele. Por uma questão de mobilidade pretendia-se também o desenvolvimento de uma aplicação remota que permitisse todas as funcionalidades da aplicação local e que corresse em qualquer dispositivo com ligação à rede e com um browser. Esta aplicação remota poderia substituir o uso de monitor, rato e teclado ligados ao Raspeberry Pi, bastando apenas este estar ligado. Outra potencialidade era o de acompanhar e controlar à distância um ensaio a decorrer.

Pretendia-se também o desenvolvimento das placas pcb necessárias para incorporar todos os componentes eletrónicos necessários ao projeto. Estas placas iriam encaixar sucessivamente em cima do Raspberry Pi.

1.3 Organização do Documento

Este documento está dividido em 4 capítulos com vista a facilitar a sua compreensão. A disposição dos capítulos é a seguinte.

- Capítulo 1: O primeiro capítulo serve para fazer uma introdução ao trabalho realizado, apresentar as motivações e os objetivos e apresentar a disposição geral do

documento.

- Capítulo 2: O segundo capítulo aborda conceitos necessários para uma melhor compreensão do trabalho. Aborda temas como a física básica por detrás de um ensaio mecânico, as linguagens de programação e o sistema operativo usados no trabalho, e a Internet, cada vez mais presente no nosso dia-a-dia.

- Capítulo 3: O terceiro capítulo debruça-se sobre a implementação do trabalho. Começa por explicar o funcionamento do controlador e a sua programação, abordar o Raspberry Pi que faz todo o controlo e o hardware usado. São depois apresentadas com detalhe a aplicação local e a aplicação remota desenvolvidas neste trabalho. Por fim é apresentado o desenvolvimento de duas placas pcb para ligar ao Raspberry Pi reunindo nessas duas placas toda a eletrónica usada no projeto.

- Capítulo 4:

  O quarto capítulo consiste nas conclusões tiradas sobre o projeto e em propostas para um futuro desenvolvimento.

Capítulo 2

Revisão do Estado da Arte

2.1 Solicitações mecânicas nos materiais

Qualquer material apresenta um conjunto de propriedades mecânicas que ditam o seu comportamento quando sujeitos às mais diversas solicitações. Uma das mais simples solicitações é um carregamento axial como vemos na Figura 2.1[1].

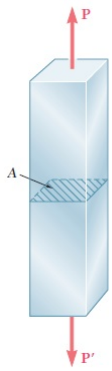

Figura 2.1: Tração uni-axial[1].

Quando aplicada a carga P, a secção "A" fica sujeita a uma tensão que é obtida pela razão entre a força e a área da secção resistente como demonstrado na Equação 2.1. Caso as forças tendam a expandir o material axialmente, designamos o carregamento por tração, caso tendam a comprimir, designamos com compressão[1].

$$\sigma = \frac{F}{A} \tag{2.1}$$

Uma força perpendicular à área resistente origina uma tensão normal como é o caso da Figura 2.1. Por sua vez uma força paralela à área resistente origina uma tensão de corte. Em termos gerais e considerando um cubo de volume infinitesimal temos a disposição de tensões como mostra a Figura 2.2[1].

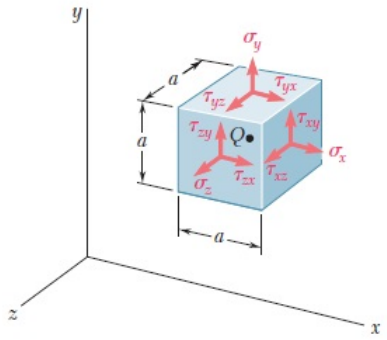

Figura 2.2: Estado geral de tenão[1].

Estas tensões induzem deformações no material. No caso de uma tração uni-axial o material alonga-se como mostra a Figura 2.3. Esta deformação é calculada como a razão entre a diferença de comprimento e o comprimento inicial como mostra a Equação 2.2[1].

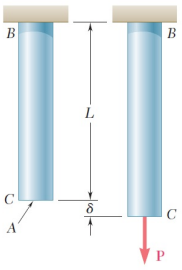

Figura 2.3: Tração aplicada ao material que resulta num alongamento[1].

$$\epsilon = \frac{l_f - l_i}{l_i} = \frac{\Delta l}{l_i} \tag{2.2}$$

A deformação é considerada elástica se o material volta à sua geometria inicial após ser retirada a carga ou plástica se a deformação imposta pela carga é permanente. Num ensaio de tração pretende-se sobretudo registar a tensão e a deformação registadas pelo material. Com estas duas grandezas é possível construir os gráficos de tensão-deformação, um dos elementos mais usados para primeira avaliação comportamental de um material[1].

Na Figura 2.4 é possivel distinguir claramente as duas regiões. A região plástica corresponde ao intervalo em que a curva é linear ou seja a derivada da tensão em ordem à deformação é constante. Esta proporcionalidade é traduzida pela Lei de Hooke que

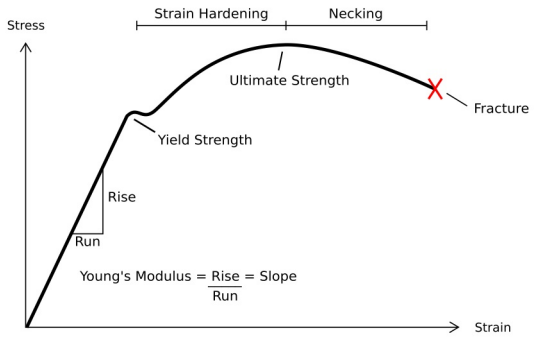

Figura 2.4: Curva Tensão-Deformação[1].

define E como o Modulo de Young demonstrada na Equação 2.3[1].

$$\sigma = E\epsilon \tag{2.3}$$

O ponto que marca o fim desta região é chamado o ponto de cedência designado no gráfico por "Yield Strength". A partir deste ponto o material deforma-se permanentemente até atingir a tensão de rutura designado no gráfico por "Ultimate Strength". Aí o material forma o chamado "pescoço" com a sucessiva diminuição da área resistente até se dar a fratura do material assinalado no gráfico por "Frature"[1].

Anteriormente vimos dois tipos de solicitações: carregamentos axiais de tração e de compressão. Outro tipo de solicitação é a flexão e o seu estudo tem um papel importantíssimo na conceção de muitas máquinas e elementos estruturais como vigas. No caso de um elemento sujeito a um par de momentos iguais em valor que atuam no mesmo plano estamos na presença de flexão pura como está ilustrado na Figura 2.5[1].

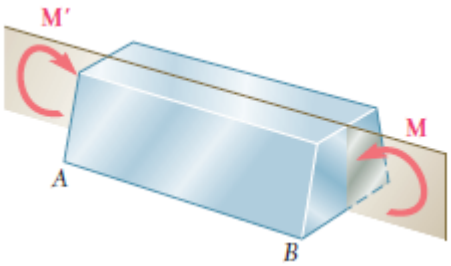

Figura 2.5: Flexão pura[1].

Um exemplo de flexão pura é mostrado na Figura 2.6 em que temos dois pontos de apoio, o ponto C e o ponto D. O valor total das reações nos pontos A e B vão ser igual ao somatório das forças nos pontos C e D uma vez que a via es´ta em equilíbrio. Como podemos ver na Figura 2.7, A força no ponto A vai ser responsável por um momento no

ponto C devido à distância que se encontra deste. O mesmo se aplica relativamente ao ponto B e D respetivamente[1].

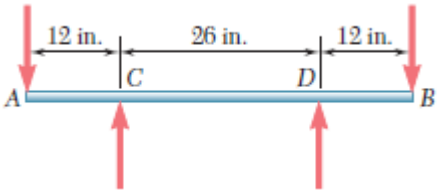

Figura 2.6: Distribuição de cargas na viga[1].

Figura 2.7: Momentos em pontos da viga[1].

Outro tipo de solicitação é a tensão de corte. Na Figura 2.8 temos uma viga encastrada e aplicamos uma carga no ponto A. Esta carga gera um momento no ponto C como vimos anteriormente mas gera também uma força de corte que dependendo do valor da área da secção transversal da viga se traduz numa determinada tensão de corte[1].

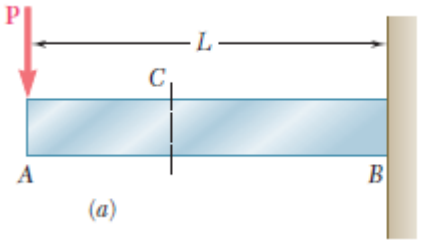

Figura 2.8: Viga encastrada com carga aplicada[1].

Nos exemplos anteriores vimos que se um material é carregado mas a tensão não excede a tensão de cedência o material retorna ao seu formato inicial quando removida a carga. Assim poderíamos concluir que qualquer que fosse o número de ciclos, desde de que não se saísse do regime plástico o material nunca se deformaria. Esta conclusão está correta para algumas dezenas e centenas de vezes, no entanto pode não se verificar quando se repetem milhares ou milhões de vezes. Quando isto acontece, ou seja a rutura do material se dá abaixo da sua tensão de rutura dá-se o nome de fadiga. Na Figura 2.10 podemos ver a relação entre a tensão aplicada e no número de ciclos necessários para se dar a rutura por fadiga tanto no alumínio como no aço.[1].

2.2 Mecatrônica

Sensores e atuadores são dois componentes principais em qualquer sistema de controlo de malha fechada. Este tipo de sistema é também designado por sistema mecatrônico.

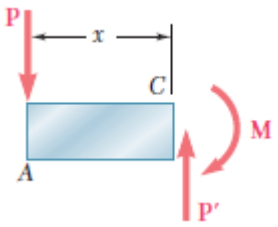

Figura 2.9: Momento e força de corte gerados no ponto C[1].

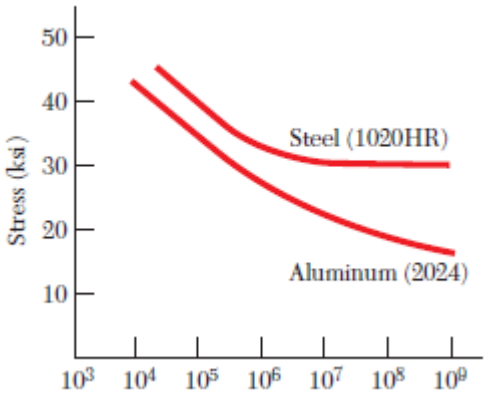

Figura 2.10: Gráfico da fadiga em aluminio e aço[1]

Tipicamente um sistema mecatrônico como ilustrado na Figura 2.11 é composto por uma unidade sensorial, um controlador e uma unidade atuadora. A unidade sensorial pode consistir simplesmente num único sensor ou ter outros componentes adicionais como filtros, amplificadores, modeladores e outros condicionadores de sinal. O controlador recebe informação da unidade sensorial, toma decisões na base de um algoritmo de controlo e fornece comandos à unidade atuadora. A unidade atuadora consiste em um ou mais atuadores e pode também conter uma fonte de alimentação e um mecanismo acoplador.[2]

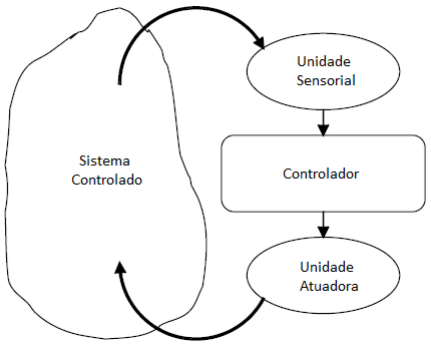

Figura 2.11: Sistema mecatrônico[2].

2.3 Sensores

Sensores e transdutores são parte essencial na instrumentação de um sistema de engenharia. Os sensores podem ser usados nestes sistemas para uma grande variedade de objetivos. Essencialmente os sensores são necessários para monitorizar e obter informação sobre o sistema. Os termos sensores e transdutores são muitas vezes usados como sinónimos. Contudo, com exatidão, um sensor sente a quantidade a ser medida enquanto um transdutor a converte numa forma que possa ser usada nas operações subsequentes.[3]

Potenciómetros. transformadores diferenciais, medidores de deformação, dispositivos piezoelétricos, medidores de caudal, termopares ou termoresistores são tudo exemplos de sensores usados na engenharia. Como exemplo de variáveis a serem medidas temos a aceleração e velocidade de um veículo, o momento numa junta robótica, a deformação num elemento estrutural, a pressão, a temperatura ou a corrente num circuito elétrico. O output de um sensor é a medição. A natureza do input e do output de um sensor são frequentemente diferentes. Enquanto um acelerómetro tem como input a aceleração, o seu output pode ser uma carga ou uma tensão elétrica. Da mesma forma o input de uma ponte que meça deformações é a deformação enquanto o seu output é uma tensão elétrica.[3]

Em qualquer um dos casos é possível calibrar o output com as unidades da grandeza a medir. O dispositivo de medição normalmente compreende duas etapas. Primeiro a grandeza a ser medida é sentida pelo elemento sensível. Depois, o sinal sentido é convertido no output. Como estas etapas ocorrem em conjunto, é normal usar-se os termos sensor e transdutor como sinónimos para designar a unidade sensor-transdutor[3].

2.4 Medição da Força

Muitos tipos de sensores de força e momento são baseados em medidores de deformação. Embora estes medidores meçam deformação, os valores medidos podem ser relacionados diretamente com tensão e a força. Os medidores de deformação também podem ser usados indiretamente para medir outras grandezas como deslocamento, aceleração, pressão ou temperatura. A variação da resistência elétrica de uma material quando deformado mecanicamente é medida pelos medidores de deformação resistivos. A resistência de um material com um comprimento l e uma área de secção A é dada pela Equação 2.4[3].

$$R = \rho \frac{l}{A} \tag{2.4}$$

Em que ρ é a resistividade do material. Partindo da equação 2.4 chegamos à equação 2.5[3].

$$\frac{dR}{R} = \frac{dp}{p} + \frac{d\frac{l}{A}}{\frac{l}{A}} \tag{2.5}$$

O primeiro termo do lado direito da equação é a variação fracional da resistividade e o segundo termo representa a variação fracional da deformação. Daí advém que a variação da resistência é dada pela variação da resistividade e pela variação da forma do material. Para deformações lineares os dois termos do lado direito da Equação 2.5 são funções lineares da deformação. A proporcionalidade constante do segundo termo em particular depende da razão de Poisson do material. Assim podemos escrever a relação descrita pela Equação 2.6[3].

$$\frac{dR}{R} = S_s e \tag{2.6}$$

A constante S_s é conhecida como "gauge factor" e o termo "e" corresponde à deformação do material. O valor de S está compreendido entre 2 e 6 para materiais metálicos, materiais esses que têm o nome de extensómetros. A variação da resistência do extensómetro que vai determinar a deformação são medidos recorrendo a um circuito elétrico adequado tipicamente um circuito ponte. Um destes tipos de circuito é a ponte de Wheatstone representada na figura 2.12. Uma ou mais das quatro resistências podem ser extensómetros.[3]

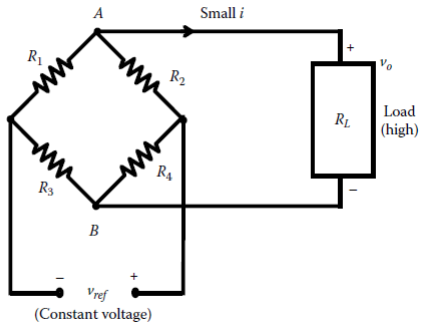

Figura 2.12: Ponte de wheatstone[3].

A tensão de saída é dada pela equação seguinte equação[3]:

$$V_{out} = [\frac{R_1}{R_1 + R_2} - \frac{R_3}{R_3 + R_4}]V_{ref} \tag{2.7}$$

A ponte diz-se em equilíbrio quando se verifica a seguinte condição[3]:

$$\frac{R_1}{R_2} = \frac{R_3}{R_4} \tag{2.8}$$

No caso da ponte se encontrar em equilíbrio, V_{out} é zero. Ao ser aplicada uma deformação a ponte deixa de estar em equilíbrio e V_{out} passa a ser diferente de zero. É importante verificar que para pequenas variações de resistência V_{out} é proporcional à variação da mesma. A partir daqui é possível calcular a deformação dos extensómetros em causa que por sua vez está relacionada com a carga imposta. [3]

2.5 Medição da Posição

Qualquer trandutor que gere uma leitura codificada(digital) pode ser considerado um encoder. Encoders de rotação são trandutores digitais que permitem medir deslocamentos e velocidades angulares. A sua alta resolução, a sua alta precisão e a sua fácil implementação assim como o facto de serem económicos são as principais vantagens de um transdutor digital, em particular um encoder de rotação.[3]

Os encoders de rotação podem ser divididos em dois grupos de acordo com a sua natureza e a interpretação do seu output. São eles os encoders incrementais e os encoders absolutos.[3]

Num encoder incremental, Figura 2.13, o seu output é um sinal pulsado, que é gerado quando o disco do transdutor roda como resultado do movimento a ser medido. Contando os pulsos e tendo em conta a cadência a que ocorrem, tanto o deslocamento angular como a velocidade angular podem ser determinados. Com um encoder incremental, o deslocamento é obtido em relação a um ponto de referência. Este ponto de referência pode ser a posição inicial do componente que se move ou um ponto de referência no disco do encoder assinalado pela emissão de um pulso index na respetiva localização no disco. Para além disso através do pulso index é possível determinar o número total de voltas efetuadas. Normalmente existe um outro pulso que permite calcular o sentido da rotação. Assim um encoder incremental normalmente inclui os sinais A, B e o Z ou index sendo que o pulso A e o pulso B emitem na mesma faixa do disco mas encontram-se desfasados num certo ângulo, o que permite através o pulso B saber o sentido de rotação. [3]

Por outro lado um encoder absoluto, ilustrado na Figura 2.14, contem no seu disco faixas de pulsos. Quando o disco roda, um resultado binário(0 ou 1), com o número de dígitos iguais às faixas no disco é gerado. Para este efeito usa-se o código de Gray. Assim em cada instante é possível ter a posição exata do disco codificada. Encoders absolutos são usados normalmente em frações de uma rotação mas também podem ser usados em várias rotações usando uma faixa adicional que emite pulsos index a cada rotação tal que acontece nos encoders incrementais.[3]

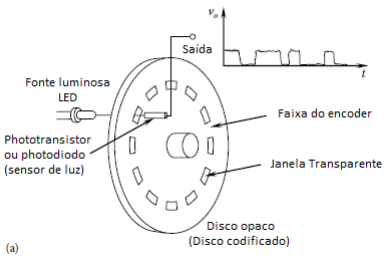

Figura 2.13: Exemplo de um encoder incremental[3].

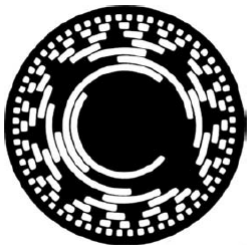

Figura 2.14: Exemplo do disco um encoder absoluto de 8-bit usando código de Gray[2].

2.6 Atuadores

Os atuadores são basicamente a força motriz por detrás de um sistema mecatrônico que aceita um comando de controlo(tipicamente sob a forma de um sinal elétrico) e provoca uma mudança no sistema controlado através da criação de uma força, de um deslocamento, calor, caudal entre outros. Tipicamente os atuadores operam conjuntamente com uma fonte de alimentação e um mecanismo acoplador como ilustrado na Figura 2.15[2].

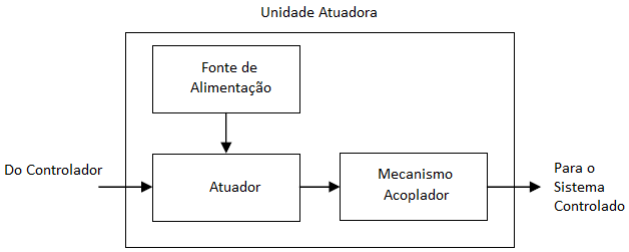

Figura 2.15: Unidade Atuadora[2].

A fonte de alimentação oferece uma determinada tensão AC ou DC a uma dada corrente máxima. O mecanismo acoplador atua como uma interface entre o atuador e o sistema físico. Os atuadores podem ser classificados consoante o tipo de energia. Eles são essencialmente elétricos, eletromecânicos, eletromagnéticos, pneumáticos ou hidráulicos. Os atuadores também podem ser classificados como binários ou contínuos consoante o número de estados da saída. Como exemplo de de um atuador binário temos um relay que possui dois estado[2].

Os atuadores elétricos assim como os pneumáticos são a principal escolha quando

estamos perante uma ação de controlo on/off. Dispositivos interruptores como diodos, transistores, triacs, MOSFETs e relés aceitam um sinal de comando de baixa energia do controlador e ligam ou desligam dispositivos elétricos como por exemplo motores, válvulas ou elementos de aquecimento. Por exemplo um interruptor MOSFET é mostrado na Figura 2.16[2]

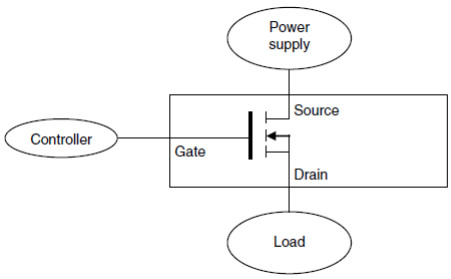

Figura 2.16: Exemplo de um MOSFET[2].

O gate do MOSFET recebe o sinal de baixa tensão vindo do controlador que permite ou não a conexão entre a fonte de alimentação e a carga.[2]

O tipo mais comum de uma atuador eletromecânico é um motor que converte energia elétrica em energia mecânica e são os atuadores mais usados na industria para fazer essa conversão. Em geral podem ser classificados como motores DC, AC ou motores de passo. Motores de corrente continua operam com tensão DC e a sua velocidade pode facilmente ser controlado fazendo variar a tensão. Eles podem ser usados desde situações onde é necessário grande potencia até situações que necessitem de uma potência mínima. Contudo os motores DC são mais caros e requerem maior manutenção quando comparados com os motores AC. Os motores AC são os mais comum uma vez que usa a corrente alterna standard, não requerem escovas ou comutadores e são por isso mais económicos. Estes motores podem ser classificados com como motores de indução, motores síncronos e motores universais de acordo com a a sua estrutura física. Um motor de indução é simples, robusto e não necessita de manutenção. Estes motores estão disponíveis em muitos tamanhos e formas consoante o números de fases usadas. Um motor de três fases é usado em situações que requerem grande potência. Um servomotor de duas fases é usado maioritariamente em sistemas de controlo de posição. Motores de indução de uma única fase são usados em muitas aplicações domesticas. Um motor síncrono é um dos motores mais eficientes na industria e é muitas vezes usado com vista a reduzir despesas. Além disso, um motor síncrono roda com uma velocidade sincronizada e é usado em aplicações que requerem operações sincronizadas. Um motor universal opera tanto com alimentação DC ou AC e é normalmente usado em situações que requerem pouca potência. O motor universal DC tem a mais alta relação potência por quilo mas tem um tempo de vida relativamente pequeno.[2]

O solenoíde é o tipo mais comum de um atuador eletromagnético. Solenoides de corrente contínua consistem num núcleo de ferro dentro duma bobine que transporta corrente. Quando a bobine é energizada, forma-se um campo eletromagnético que cria a força necessária para mover o núcleo de ferro.[2]

Atuadores hidráulicos e pneumáticos são normalmente motores rotativos, pistões lineares ou válvulas de controlo. Atuadores pneumáticos usam ar sob pressão e são mais

frequentes em situações que requerem de pouca a média força, pequenos deslocamentos e alta velocidade de atuação. Os Atuadores hidráulicos usam óleo pressurizado incompressível. Estes atuadores permitem a produção de enormes forças juntamente com grandes deslocamentos de uma maneira rentável economicamente. A principal desvantagem dos atuadores hidráulicos é que são mais complexos e necessitam de uma maior manutenção. Os motores rotativos são normalmente usados em aplicações onde são necessários baixas velocidades e grandes momentos. Os atuadores do tipo pistão encaixam em aplicações onde se deseja um deslocamento linear. As válvulas de controlo são usadas para fazer o controlo do caudal de um fluido.[2]

2.7 Mosfet

Os Mosfet são uma classe de transístores cujo funcionamento assenta no efeito do campo elétrico daí o nome "metal-oxide-semiconductor-field-effect-transistor". Esta classe de mosfets é substancialmente diferente dos transístor de junção. O conceito que está na base do funcionamento destes transístores é o facto da espessura do canal condutor poder ser controlado pela aplicação dum campo eletromagnético daí um mosfet se comportar como uma resistência cujo valor é controlado por uma tensão. Existem três grandes grupos dentro dos mosfets: "Enhanced Mode Mosfets" e "Depletion-Mode Mosfets" e JFETS. Cada um destes grupos pode ser fabricado como um dispositivo "n-channel" ou "p-channel" onde o "p" e o "n" indicam a natureza da dopagem no canal semicondutor. Os mosfets destes três grupos têm um comportamento muito idêntico. Num mosfet temos três terminais. O gate, análogo à base num BJT(transistor de junção) o drain análogo ao colector e o source análogo ao emissor. No caso de um mosfet "n-channel Enhanced Mode", ao ser aplicada uma tensão ao gate acima de uma certo valor respetivamente à source, o mosfet estabelece a ligação entre o drain e a source. Abaixo desta tensão o mosfet corta essa ligação

2.8 Máquina de Ensaios

O carregamento mais natural numa máquina de ensaios linear é a tração uni-axial, mas podem ser impostos outros carregamentos como a compressão, o corte ou a flexão. Uma máquina de ensaios é constituída por estes 3 componentes principais: o atuador, a leitura da força aplicada e a leitura do deslocamento registado[4].

Na Figura 2.17 está representado um provete típico usado num ensaio de tração. As zonas finais ou ombros têm uma maior largura para permitir a fixação. A zona importante do provete é a zona descrita na Figura 2.17 como comprimento útil. A área da secção transversal desta região é mais reduzida para garantir que deformação e a rutura do material se dá nesta zona. A região do comprimento útil é a região na qual as medições são feitas e encontra-se centrada na zona mais estreita[4].

As máquinas de ensaio podem ser eletromecânicas ou hidráulicas. A principal diferença é a forma como a carga é aplicada. As máquinas de ensaios eletromecânicas baseiam-se na velocidade variável do motor elétrico, num sistema redutor de engrenagens e um, dois ou quatro sem-fins que permitem a movimentação no sentido ascendente e descendente. Este deslocamento impõe respetivamente tensão e compressão ao provete. A velocidade deste deslocamento pode ser variada, variando a velocidade do motor elé-

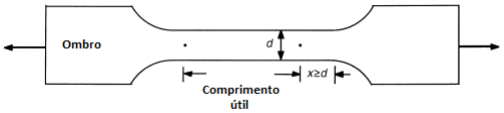

Figura 2.17: Provete usado num ensaio de tracção[4].

trico. Um servo sistema de malha fechado controlado por um microprocessador pode ser implementado para um controlo preciso da força a uma dada velocidade pretendida. Na Figura 2.18 podemos ver o esquema básico de uma máquina eletromecânica[4].

As máquinas de ensaio hidráulicas são baseadas na atuação de um ou dois pistões que permitem o movimento ascendente e descendente. Na máquina hidráulica de controlo manual, o operador ajusta a carga através de uma válvula de pressão. Num servo sistema hidráulico de malha fechada este ajuste é feito por uma servo válvula acionada eletricamente o que permite um controlo preciso. Em geral, as máquinas eletromecânicas são capazes de operar numa maior janela de velocidades e maior deslocamentos enquanto as máquinas hidráulicas são mais económicas quando se pretende gerar cargas mais elevadas[4].

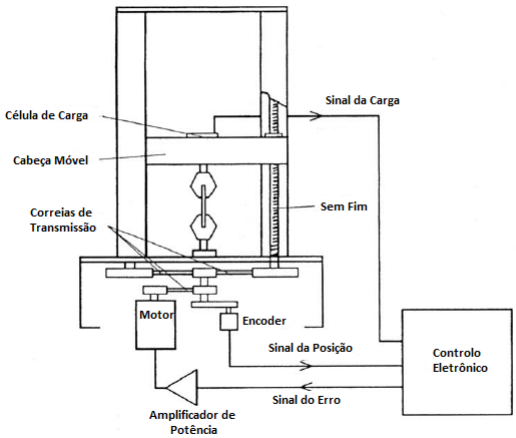

Figura 2.18: Exemplo de uma máquina de ensaios eletromecânica [5].

2.9 Raspberry Pi

Raspberry Pi é um pequeno "single-board computer" desenvolvido no Reino Unido pela "Raspberry Pi Foundation", uma fundação sem fins lucrativos. Isto segnifica que é um computador normal mas confinado numa única placa impressa mais ou menos do tamanho de um cartão de crédito. O Raspberry Pi foi desenvolvido com o intuito de servir como material de estudo na ciência computacional. Rapidamente se tornou popular e o seu uso estende-se de casas, salas de aulas, escritórios, fábrica e até barcos auto-pilotados[6].

2.10 A Internet

Até ao Inicio de 1990 a Internet era usada maioritariamente por investigadores, académicos e estudantes universitários. Este uso incluía o "log in" em hosts remotos, a transferência de ficheiros e o envio e receção de emails assim como de noticias. Por esta altura a Internet era praticamente desconhecida fora do mundo académico e comunidades de investigação. É então nesta altura que surge a "World Wide Web". A Web foi a primeira aplicação da Internet que captou a atenção do publico em geral. Ela mudou drasticamente a forma como interagimos. Talvez o que mais atrai os utilizadores da Web seja o facto de ela funcionar "on demand", isto é os utilizadores recebem o que querem, quando querem. Este conceito contrasta com os meios tradicionais de transmissão como o radio e a televisão que obriga os utilizadores a estarem sintonizados quando o conteúdo é transmitido. Outro aspeto que torna a Web apelativa é a facilidade com que qualquer utilizador pode publicar informação na Web a um custo extremamente baixo. "Hyperlinks" e motores de busca permitem-nos navegar nesta imensidão de "Websites" disponíveis na Web[7].

A Internet é um sistema extremamente complexo constituído por inúmeras aplicações, protocolos e vários tipos de sistemas ligados entre si através de diversos tipos de comunicação. Uma maneira de simplificar esta representação é estruturar a Internet por camadas de protocolos como mostra a Figura 2.19 que ilustra o modelo OSI. Cada camada compreende protocolos assim como o software e o hardware que os implementam. Uma camada pode ser vista como um conjunto de serviços que esta oferece à camada acima na hierarquia. Cada camada disponibiliza estes serviços através de ações dentro da própria camada ou usando serviços oferecidos pela camada imediatamente abaixo na hierarquia.

Cada protocolo pode ser implementado em software, em hardware ou numa combinação dos dois[7].

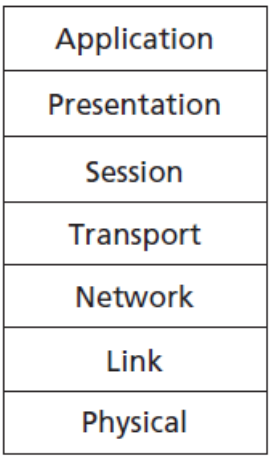

Figura 2.19: Camadas da Rede[7].

A camada da aplicação é onde as aplicações de rede e os protocolos por elas usados se situam. Esta camada inclui muitos protocolos como o protocolo HTTP (destinado ao pedido e transferência de documentos Web), SMTP (destino à transferência de mensagens

email) e FTP (destinado à transferência de ficheiros entre dois sistemas). Também é o caso do protocolo DNS que permite a tradução de nomes mais facilmente entendidos por humanos para endereços de rede de 32-bit. Um protocolo desta camada é um protocolo usado por duas aplicações em dois sistemas ligados à rede que o usam para trocar pacotes de informação entre si[7].

O protocolo HTTP de HyperText Transfer Protocol é o cerne da Web e está definido em [RFC 1945] e [RFC 2616]. O HTTP é implementado em dois programas: o programa cliente e o programa servidor. Este dois programas executam em diferentes sistemas e comunicam entre si trocando mensagens HTTP. O protocolo HTTP define a estrutura destas mensagens e como o cliente e o servidor as devem trocar. Uma pagina Web é constituída por objetos. Um objeto pode ser um ficheiro HTML, uma imagem JPEG, um applet JAVA ou um clip de vídeo que é endereçado através de um URL único. A maior parte das páginas Web consistem num ficheiro HTML base e vários objetos referenciados por ele. O ficheiro base HTML referencia os outros objetos através do URL destes objetos. Cada URL é dividido em duas partes: o hostname do servidor em que o objeto está alojado e o caminho para o objeto. Como exemplo de programas do lado do cliente que usem o protocolo HTTP temos os famosos browsers que nos permitem navegar na Internet como por exemplo o Chrome, Firefox, Safari, Microsoft Edge entre outros. Do lado do servidor temos os chamados servidores Web como por exemplo o Apache e o Microsoft Internet Information Server. Este servidores Web que implementam o HTTP alocam objetos referenciados por um URL único e este protocolo define como se dá a transferência destes objetos para o cliente. Resumidamente o servidor recebe pedidos e responde com mensagens HTTP que contêm os objetos[7].

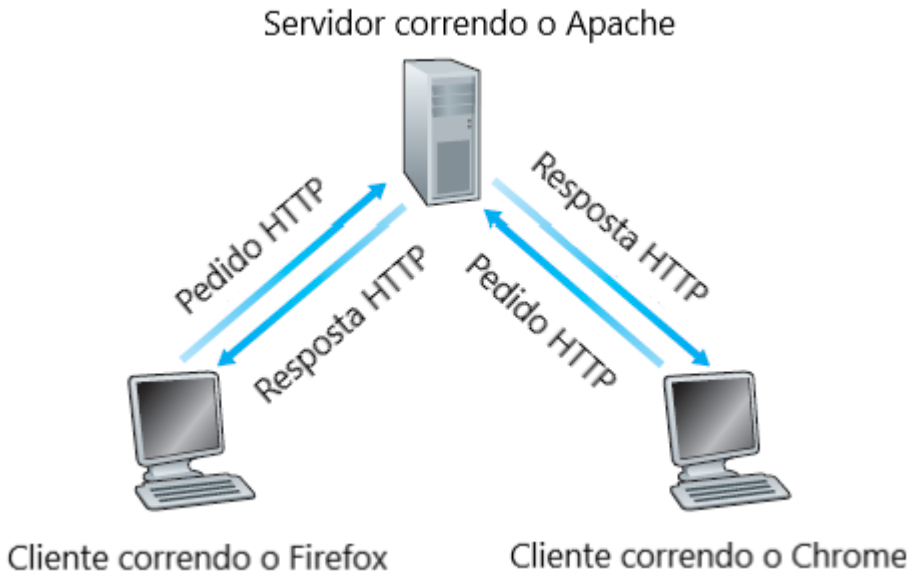

Figura 2.20: Exemplo de uma pedido e resposta usando o protocolo HTTP[7].

O protocolo HTTP assenta no protocolo TCP pertencente à camada imediatamente abaixo. Outro protocolo dessa camada é o UDP. A principal diferença entre os dois é que

o protocolo TCP garante que informação chega ao destinatário, ao contrário do protocolo UDP. Por outro lado o UDP privilegia sobretudo a velocidade e é útil em situações em que a cadência com que essa informação chega é mais importante do que a garantia que essa informação chega na totalidade como é o caso de transmissões em tempo-real. No caso do HTTP o principal requisito é que a informação seja transferida integralmente. Aqui fica evidente as vantagens da divisão em camadas uma vez que o protocolo HTTP não tem de se preocupar com os processos envolvidos na transferência de informação ficando estes ao encargo dos protocolos nas camadas mais abaixo na hierarquia. Outra característica do HTTP é que este não guarda qualquer informação dos clientes e é por isso chamado um protocolo "stateless". Normalmente um servidor Web encontra-se sempre ligado num IP fixo e pode servir potencialmente milhões de clientes[7].

De seguida são apresentados alguns dos principais tipos de pedidos do protocolo HTTP:

- GET
- POST
- PUT
- HEAD
- DELETE

A Figura 2.21 ilustra as várias camadas usadas ao longo de um transmissão de informação na rede.

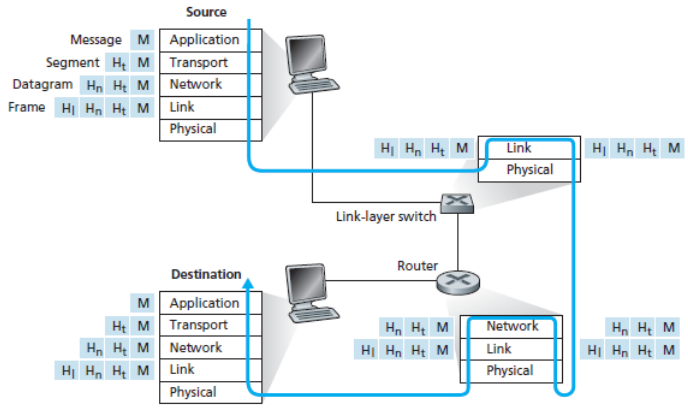

Figura 2.21: Transmissão nas várias camadas de rede[7].

2.11 A linguagem C e o Linux

A linguagem de programação C está intrinsecamente ligada ao sistema operativo Unix. O C foi inventado por Dennis Ritchie em 1972 quando ele e Ken Thompson desenvolviam este sistema operativo. O C foi fortemente influenciado pela linguagem B que tinha sido inventada anteriormente por Ken Thompson. O C foi criado com o intuito de ser

uma ferramenta para os programadores e portanto o seu principal objetivo é ser uma linguagem útil e eficiente.[10] Outras linguagens tiveram na sua génese outros objetivos como por exemplo Pascal que pretendia ser uma linguagem em que se pudesse ensinar boas práticas de programação ou BASIC que pretendia relembrar o inglês e assim ser facilmente aprendida por estudantes pouco familiarizados com a computação. O desenvolvimento do C como uma linguagens desenhada para os programadores tornou-a uma das linguagens de eleição dos tempos modernos. Nestas últimas 4 décadas o C tem-se distinguido como uma das mais importantes e populares linguagens de programação. Cresceu sobretudo porque as pessoas que o experimentam gostam. Entretanto apareceram linguagens de mais alto nível como C++, Java ou Objective-C que conquistaram o seu espaço mas o C continua a ter um papel importante sobretudo em programação de baixo-nível como sistemas operativos ou sistemas embutidos. Algumas das virtudes do C podem ser destacadas[10]:

- Eficiência
- Código compacto
- Portabilidade entre sistemas

O C é eficiente pois tira o máximo partido da arquitetura em causa. Os programas em C tendem a ser compactos e a executar com rapidez. O C exibe um controlo mais fino geralmente associado com uma linguagem assembly (uma linguagem assembly é a representação por mnemónica do conjunto de instruções usadas por um determinado CPU. Cada CPU tem a sua respetiva linguagem assembly). Um destes aspetos é o fato do C permitir a manipulação da memória o que se traduz em programas mais eficientes. A portabilidade do C implica que um programa escrito para uma arquitetura pode ser portado para outra arquitetura com pouca ou nenhuma modificação requerendo somente que exista um compilador para essa arquitetura. Há várias linguagens que tentam ser portáteis mas o C continua a dominar nesta área. Tipicamente as partes dos programas que não são portáteis são o acesso a um dispositivo particular de hardware como um monitor ou uma funcionalidade oferecida por um sistema operativo específico. Por causa relação entre C e Unix, os sistemas Unix tipicamente trazem um compilador C por defeito. As distribuições Linux normalmente trazem o compilador gcc da GNU. Resumindo, quer estejamos a usar um computador pessoal, uma "workstation profissional" ou um "mainframe" as probabilidades são grandes que exista um compilador de linguagem C para esse sistema. O C é uma linguagem poderosa e flexível. Por exemplo, a maioria do sistema operativo Unix foi escrita em C. Muitos compiladores e interpretadores para outras linguagem são escritos em C como por exemplo FORTRAN, Perl, Python, Pascal, LISP entre outros. Ao longo da história o C tem sido usado para resolver problemas de física e engenharia e até para produzir efeitos especiais para filmes.[10]

O C foi desenhado para suprir as necessidades dos programadores. Permite ao programador aceder ao hardware e permite a manipulação individual dos bits na memória. O C não é tão restritivo em termos do que o programador pode fazer como por exemplo Pascal ou mesmo C++. Esta flexibilidade é ao mesmo tempo uma vantagem e um perigo. A vantagem é que em muitas situações, como por exemplo a conversão entre formatos de dados é mais simples em C. O perigo é que em C se pode cometer erros que são impossíveis noutras linguagens. Ou seja, traz mais liberdade mas ao mesmo tempo

deposita no programador maior responsabilidade. Ao mesmo tempo, o C também tem em quase todas as suas implementações uma biblioteca rica e extensa que vai ao encontro das necessidades do programador.[10]

O sistema operativo Linux está intimamente ligado à linguagem de programação C uma vez que praticamente todo o seu Kernel é escrito em C excetuando algumas partes que dependem da arquitetura e por isso são escritas em Assembly. Os primeiros passos na produção do Linux começaram no verão de 1991, mas antes de existir o Linux, existia o GNU project. Este projeto tinha começado cerca de uma década antes e tinha como objetivo a produção de componentes de software grátis para assim conseguir criar um sistema operativo totalmente grátis como é hoje em dia o Linux. Sem o projeto GNU talvez o Linux nunca tivesses existido e sem o Linux o projeto GNU possivelmente não teria a visibilidade que tem hoje. Ambos os projetos se complementaram e beneficiaram com isso.[11] O objetivo inicial do projeto GNU era criar um sistema operativo ao estilo do UNIX com todas as ferramentas necessárias. Demorou cerca de uma década até estarem prontas essas ferramentas como o compilador C GCC, o editor de texto da GNU emacs e muitas outras ferramentas. O projeto teve grande sucesso inicial mas ao longo dos anos 80 faltou-lhe um componente essencial: o Kernel. Como não tinha Kernel próprio servia para os utilizadores instalarem as suas ferramentas em sistemas operativos de uso comercial como o UNIX. O projeto nunca se sentiu completo seu um Kernel próprio. Houve tentativas de usar o microkernel HURD assim como o microkernel Minix escrito pelo professor Andrew Tanenbaum mas por uma ou outra razão estas soluções não foram consideradas aptas na altura em que apareceu o Linux. Entretanto, um jovem estudante Filandês chamado Linus Torvalds, que estudava na Universidade de Helsinquia sentiu que o Minix tinha demasiadas lacunas e então começou a trabalhar no seu próprio sistema operativo desenhado especificamente para o seu micro computador AT-386. Assim nasceu o Linux e apesar da modestia inicial de Linus, o interesse no Kernel do Linux cresceu rapidamente a nível mundial e atraiu muitos programadores que passaram a colaborar no projeto. Os programadores do Linux utilizaram muitas das ferramentas do projeto GNU e contribuíram com melhorias para as existentes. [11]

Hoje em dia, o Linux corre em arquitetura como Alpha, ARM,PowerPC,SPARC, x86-64 entre muitas outras. Corre desde sistemas tão como simples relógios de pulso até supercomputadores do tamanho de salas. Embora o Linux tenha adotado muitas das ideias do Unix e implemente a sua API, o Linux não é um descendente direto do UNIX como o são outros sistemas operativos. [12] Um dos mais interessantes aspetos do Linux é que este não é um produto comercial mas um projeto de colaboração desenvolvido através da Internet. Embora Linus se mantenha como criador e responsável pelo Kernel, o progresso é assegurado por um grupo vasto de colaboradores. Qualquer programador pode contribuir para o desenvolvimento do Linux. O Kernel, assim como grande parte do sistema é software grátis e "open source". O Kernel do Linux está licenciado sob o GNU General Public License (GPL) version 2.0. Consequentemente qualquer pessoa pode fazer download do "source code" e fazer as modificações que entender. A única restrição é que o código modificado continue a ser licenciado da mesma maneira incluindo a disponibilização do seu "source code". A base de um sistema Linux é constituído pelo Kernel, pela biblioteca standard do C, compiladores e outras ferramentas utilitárias básicas como o processo de login e a shell. O sistema Linux moderno também inclui habitualmente um sistema gráfico Window X assim como um "desktop enviroment" como o GNOME.

Qualquer sistema operativo oferece uma API de systemcalls contra a qual é possível programar e assim usar as funcionalidades oferecidas por esse sistema operativo. No caso de um sistema Linux, este vem por defeito com o libc, a biblioteca standard C do GNU que oferece uma serie de rotinas que facilitam a vida do programador assim como vários wrappers à volta de systemcalls como mostra a Figura2.22

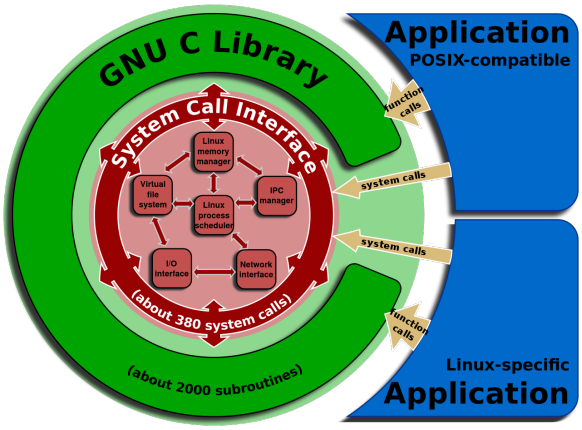

Figura 2.22: Libc e o Linux.

2.12 As aplicações Web, o Javascript e o PHP

Uma aplicação Web é composta por vários componentes que trabalham em conjunto. Como clientes temos os computadores, tablets ou smartphones que usam a aplicação Web. Eles acedem às páginas Web através de um browser. O servidor contém os ficheiros que compõem a aplicação Web. Uma rede é aquilo que permite os clientes e os servidores comunicarem e pode ser um rede local denominada LAN (Local Area Network) ou a Internet que já foi abordada anteriormente e que é um aglomerado gigante de várias redes. Uma página estática é uma página que não muda entre diferentes pedidos. Este tipo de página é enviado diretamente do Web Server para o Web browser que a pediu. Uma página estática pode ser reconhecida pela extensão que apresenta. Se a extensão for .htm ou .html é provavelmente uma página estática. O processo inicia com o cliente a fazer o pedido HTTP ao Web Server. O Web Server ao receber o pedido, lê o ficheiro do disco e devolve-o como parte da resposta HTTP. O browser ao receber a resposta HTTP, traduz e renderiza a página html mostrando no browser. Quando um utilizador carrega no link ou digita outra endereço web no barra de endereço do browser o processo repete-se.

Por outro lado, uma página dinâmica é uma página criada por um programa ou script no lado do Web server cada vez que é feito um pedido. Este programa ou script é executado baseado na informação que é enviada juntamente com o pedido HTTP. Baseado na extensão da webpage, o Web server determina se é ou não uma página dinâmica e caso seja, carrega o programa ou script do disco, executa-o e compoe assim o ficheiro html que devolve ao cliente.[8]

No caso de ser um script PHP o Web Server lança o interpretador PHP que vai processar o script. A seguir são apresentados os passos na criação de uma página web

dinâmica em que o Web Server consulta uma base de dados:[9]

- o Utilizador insere o endereço na barra de endereço do browser do género http://www.server.com.
- O browser procura na Internet o IP do endereço server.com
- O browser faz um pedido HTTP para adquirir a homepage do Web Server cujo IP foi encontrado no passo anterior.
- O pedido viaja pela rede, que pode ser a Internet e chega ao Web Server.
- O Web Server recebe o pedido e carrega a página Web do disco.
- O Web Server apercebe-se que a página web contem partes que são script PHP e lança o interpretador PHP.
- O interpretador PHP processa o script
- O interpretador encontra instruções para ler da base de dados e executa-as.
- O interpretador PHP compõe a página web e devolve-a ao Web Server.
- O Web Server envia a página para o cliente neste caso o browser que a renderiza e mostra no ecrã.

É de assinalar que o browser processa a página web normalmente sem ter o conhecimento que ela foi criada dinamicamente[8]

O PHP permite a execução de código no lado do servidor. Da mesma forma, também podemos ter a execução de código script do lado do cliente usando por exemplo a linguagem Javascript. Hoje em dia, o Javascript é usado na maioria das páginas de um website moderno. Quando o cliente pede uma página, tanto o html como o código JavaScript a ele associado são retornados pelo servidor. O código JavaScript é então executado no lado do cliente pelo JavaScript engine do browser e permite uma grande dinamização da página Web em questão. Ao longo do tempo, os programadores têm desenvolvido bibliotecas que contêm código para as funções mais comuns. A mais popular dessas bibliotecas é o JQuery[8].

JSON(JavaScript Object Notation) é um dos vários formatos para transferir dados. Quando pretendemos transferir os dados enviando em serie os vários elementos, estes são convertidos para uma certo formato e a este processo dá-se o nome de serialização. Quando queremos trabalhar novamente com os dados, estes são convertidos para a sua estrutura original e temos o processo de desserialização. Estes formatos pode ser do tipo binário ou textual. O JSON é do tipo textual assim como é o XML ou CSV. Como o JSON é um subset do Javascript é a forma preferencial para transferir ou guardar dados quando programamos em Javascript. Na Figura 2.23 podemos ver a arquitectura de um interpretador php e de um interpretador javascript[8].

2.13 Base de Dados

Um sistema de gerenciamento de base de dados ou DBMS(database-management system) é um conjunto de dados relacionados entre si e de um conjunto de programas que

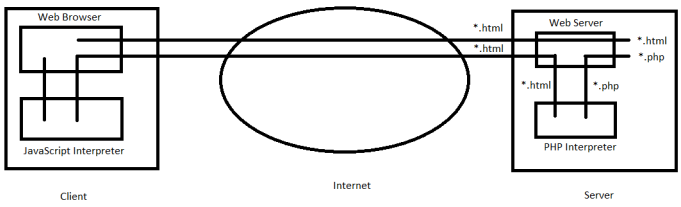

Figura 2.23: Esquema do interpretador javascript e do interpretador php.

permitem aceder a esses dados. Ao conjunto de dados geralmente chama-se de base de dados. O principal objetivo de um DMBS é proporcionar uma forma de guardar e obter dados de uma forma conveniente e eficiente. Uma base de dados tem como objetivo guardar grande quantidade de informação. A sua gestão involve definir estrutura para guardar esta informação e mecanismos para a manipular. Para além disso tem de assegurar a integridade e segurança ada informação no caso de um crash ou de um acesso não autorizado[13].

Um sistema de base de dados proporciona uma linguagem de definição e uma linguagem de manipulação para expressar queries e updates. Na prática estas duas linguagens estão agrupadas numa só como é o caso da linguagem SQL muito difundida nos dias de hoje. Esta linguagem permite os seguintes tipos de acesso[13]:

- Obter informação guardada na base de dados
- Inserir nova informação na base de dados
- Eliminar informação da base de Dados
- Modificar informação guardada na base de dados

Existem dois tipos de linguagens que permitem a manipulação de uma base de dados: linguagens processuais e linguagens declarativas[13].

Numa linguagem processual o programador tem de definir os dados a obter assim como as ações necessárias para os obter[13].

Uma linguagem declarativa requer que o programador especifique os dados mas não necessita de especificar a forma de os obter. As linguagens declarativas são normalmente mais fáceis de aprender do que uma linguagem processual. Como o programador não especifica a forma de obter os dados, cabe ao sistema da base de dados a implementação de uma maneira eficiente de aceder aos mesmos[13].

Uma query é uma frase que requer à base de dados retorno de informação. O SQL é um exemplo de uma linguagem de base de dados query. As query introduzem um nível de abstração em relação à definição e estrutura de dados mas também à sua manipulação. Cabe à parte do sistema da base de dados responsável pelo processamento das querys traduzir este pedidos em sequências de ações ao nível físico da base de dados. Uma query tem como input uma ou mais tabelas e devolve sempre uma tabela[13].

MariaDB é um dos servidores de base de dados mais comuns no mundo. Foi criado pelos fundadores do MySQL e tem como objetivo permanecer Open Source. Utilizadores

conhecidos incluem a Wikipedia, o Google e a WordPress.com entre outros. MariaDB é usada porque é rápida, escalável, robusta e versátil podendo ser usada numa grande variedades de situações. Este software oferece uma interface SQL para aceder aos dados. As últimas versões do MariaDB incorpora funcionalidades GIS e JSON.[14]

Capítulo 3

Implementação

3.1 Hardware Usado

Neste projeto o Raspberry Pi é responsável pelo controlo da máquina de ensaios, pela leitura da posição e da força e pela conexão com a base de dados e a aplicação remota. A comunicação com o controlador LCC-11 é feita através do protocolo RS232. A leitura da posição é feita através do protocolo SPI entre o Raspberry Pi e o encoder LS7266R. Quanto à leitura da força, esta é feita pelo módulo HX711 através do uma comunicação em serie idêntica ao I2C. As características destes componentes são apresentadas mais à frente.

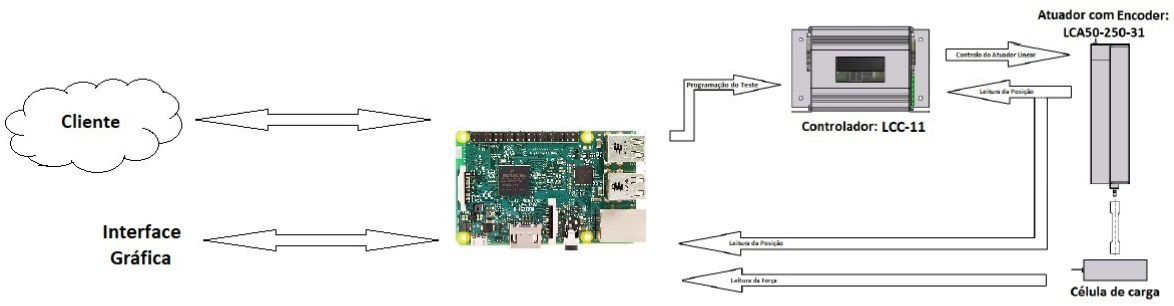

Figura 3.1: Esquema dos principais elementos do projeto.

3.1.1 O controlador LCC-11

Um controlador de movimento é um dispositivo eletrónico com software incorporado que é responsável pelo controlo preciso e eficiente dos movimentos de um motor elétrico. Esse controlo está esquematizado na Figura 3.4[16].

O LCC-11 é um controlador de movimento da SMAC. É um controlador de eixo único para motores sem escovas ou servo motores. Permite modos de operação como posição, velocidade, torque ou homing. O homing pode ser feito recorrendo ao limite físico ou a um

index. O gerador de trajetórias é do tipo trapezoidal e o feedback da posição é adquirido lendo um encoder incremental até 2 milhões de incrementos por segundo. É alimentado de 24-48V DC e contem uma entrada analógica de 0-5V e uma saída analógica de 0-10V com um DAC com 16 bits de precisão. O LCC-11 é em tudo igual ao controlador LCC-10 expecto nesta precisão que no caso do LCC-10 é de somente 10 bits. Contém também 4 entradas e saídas digitais TTL. Em termos de memória contém 100 registos e 16Kb de memória não volátil. Na Figura 3.2 podemos ver as ligações elétricas assim como a ligação ao encoder. Na Figura 3.3 podemos ver a interface de comunicação. O LCC-11 é um controlador EMCL.

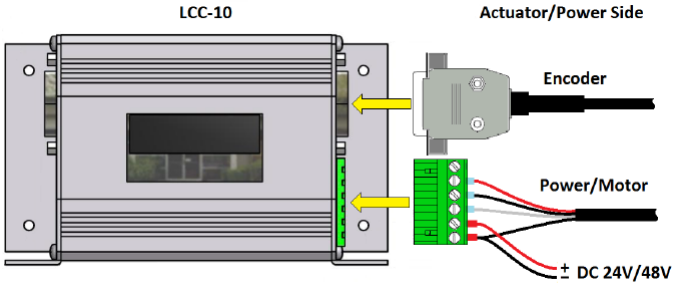

Figura 3.2: Ligações elétricas e ligação ao encoder do LCC-11[15].

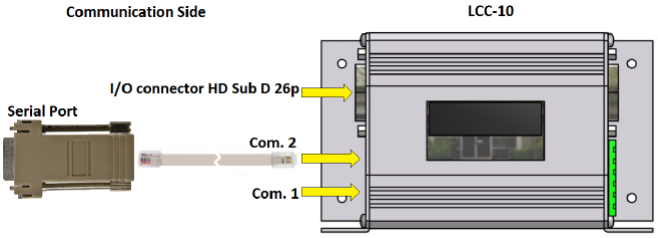

Figura 3.3: Intereface de comunicação do LCC-11[15].

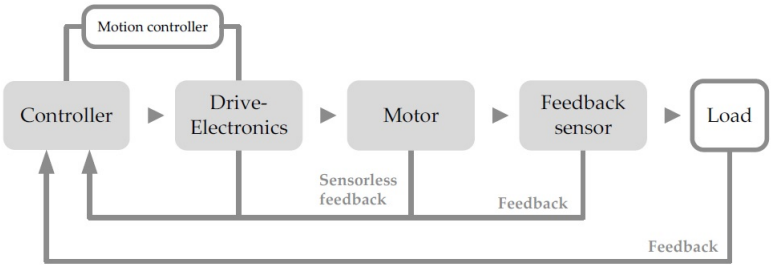

Figura 3.4: Controlo da Posição[16].

EMCL é a abreviatura de embedded motion control library e é um firmware destinado a gerir todas as funções referentes ao controlo de movimento e comunicação de um controlador de movimento. o emcl é desenhado para trabalhar com os dispositivos dsPIC33F da Microchip[16].

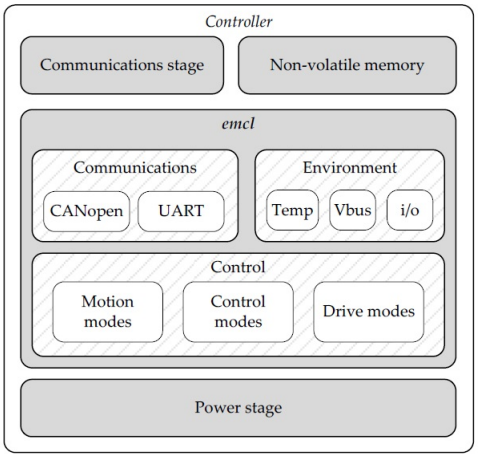

Figura 3.5: Os vários blocos que constituem um controlador emcl[16].

A Figura 3.5 mostra os blocos de hardware e software que constituem um controlador emcl. O "power stage" é responsável por converter os sinais de baixa tensão gerados pelo emcl em sinais de mais alta tensão usados para acionar o motor. Do mesmo modo, o "communications stage" é responsável por ajustar os níveis de tensão usados pelos protocolos CAN e RS232[16].

O acesso às funcionalidades e parâmetros do controlador de movimento emcl é feito exclusivamente recorrendo a objetos. Cada objeto pode ser interpretado como sendo um endereço e memória no qual é guardado um valor cujo significado depende do objeto em causa. Um objeto pode conter por exemplo a posição atual ou a velocidade imposta[16].

Os processos internos do emcl usam os objetos para executar função e para guardar resultados de operações. Os objetos podem ser modificados ou lidos usando os protocolos de comunicação CAN e RS232. Podemos considerar o emcl como sendo constituído por 3 camadas como mostra a Figura 3.6[16].

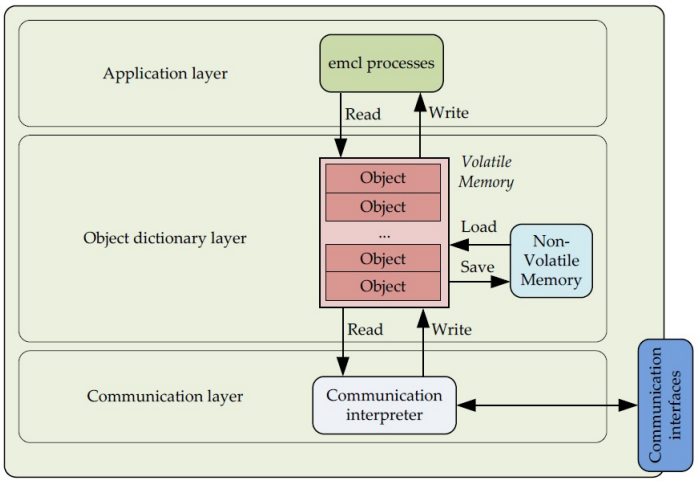

Figura 3.6: Camadas que constituem o emcl[16].

Alguns objetos podem ser guardados na memória não volátil do sistema (NVM) permitindo reter o valor. Esta funcionalidade permite não ser preciso configurar objetos cada vez que o sistema é iniciado, pois são iniciados com os valores presente na NVM. Cada objeto também chamado de "register" é definido pelos seguintes campos[16]:

Index / SubIndex	*Specifies the object number.*
Name	*Specifies the object name.*
Object type	*Specifies the object type.*
Access	*Specifies the type of object access.*
Data type	*Specifies the data type.*
PDO mapping	*Specifies if object could be accessed via PDO (only valid for CANopen communications).*
Value range	*Specifies the range of allowed values.*
Default value	*Specifies the default value of the object.*

Figura 3.7: Campos que definem um objeto[16].

O Index, composto por 16 bits é usado para endereçar todos os objetos do dicionário de objetos. No caso de ser uma variável simples o Index referencia o valor dessa variável diretamente. Caso se trate de um "Record" ou de um "Array" o "Index" refere-se a toda a estrutura. Neste caso é usado o "Subindex" de 8 bits para aceder a cada membro destas estruturas[16].

Podemos ter os seguintes tipos de objetos[16]:

- VAR: Indica o uso de um tipo básico (UINT8, INT16, STR, etc)
- ARRAY: Indica que a estrutura é composta pelo mesmo tipo básico como por exemplo UINT16.
- RECORD: Indica que a estrutura é composta por diversos tipos básicos.

Cada objeto permite os seguintes tipos de acesso[16]:

- Read/Write (RW): Acesso de leitura e escrita. O objeto pode ser consultado e modificado.
- Read only (RO): Acesso de leitura. O objeto pode apenas ser consultado.
- Constant (CONST): O objeto é constante e pode apenas ser consultado.
- Write only (WO): Acesso de escrita. O objeto pode apenas ser modificado.

Os tipos de dados básicos são os seguintes[16]:

- UINT8: 1 byte de tamanho. Intervalo [0 to 255].
- UINT16: 2 byte de tamanho. Intervalo [0 to 65535].

- UINT32: 4 byte de tamanho. Intervalo [0 to 4294967295].
- UINT64: 8 byte de tamanho. Intervalo [0 to 18446744073709551616].
- INT8: 1 byte de tamanho. Intervalo [-128 to 127]
- INT16: 2 byte de tamanho. Intervalo [-32768 to 32767]
- INT32: 4 byte de tamanho. Intervalo [-2147483648 to 2147483647]
- STR: Sequência de caracteres terminado pelo caracter nulo.

Os controladores emcl podem trabalhar com dois tipos de comunicações standard : CAN and RS232. Na versão CAN, o emcl usa o protocolo CANopen. CANopen é um protocolo implementado num barramento CAN e é gerido pela organização CiA[16].

Neste projeto foi usado o protocolo RS232 pelo que a versão CAN não será aprofundada.

O RS232 (também conhecido como Electronic Industries Alliance RS-232C) é uma interface que atribui regras para uma comunicação em série de dados binários entre um DTE (Data Terminal Equipment) e um DCE (Data Communication Equipment), embora a interface RS232 possa ser usada noutras situações. Esta interface é usada em distâncias pequenas na ordem dos 15 metros e com baixas velocidades de transmissão na ordem dos 20 KBps. A interface pode trabalhar como síncrona ou assíncrona, e como simplex, half duplex ou full duplex[16].

O configuração por defeito do RS232 usado no emcl é mostrada na seguinte figura[16]:

Baudrate	*115200 bps*
Data bits	*8 bits*
Parity	*None*
Stop bits	*1 bit*
Flux control	*None*

Figura 3.8: Configuração por defeito da interface RS232[16].

O Baudrate da comunicação não é fixo e pode ser alterado com o respetivo objeto[16].

Os comandos usados em comunicações RS232 pelos controladores baseados em emcl usam código ASCII. O seu formato, semelhante às comunicações CANopen contém a seguinte informação[16]:

NODE ID	SPACE	FCT	SPACE	OBJECT	SPACE	VALUE	CR
0-127	0x20(**)	'R' / 'W'	0x20	0 a 0xFFFFFF	0x20(*)(**)	(*)	0x0D(**)

Figura 3.9: Comando RS232 usado pelo emcl[16].

- NODE ID: Identifica o destinatário da mensagem. Pode tomar os valores entre 0-127. Pode ser expresso em decimal ou em hexadecimal (adicionando o prefixo '0x').

- FCT: Identifica o tipo de função a exucutar podendo ser de leitura ou escrita. Quando se trata de leitura o campo FCT contém o carater 'R', caso se trate de escrita contém o caracter 'W'.
- Object: Contém o número do objeto em causa. Este número é construido com a junção do 'SubIndex' e do 'Index'. O 'SubIndex' precede o 'Index'. Pode ser expresso em decimal ou hexadecimal (adicionando o prefixo '0x').
- Value: Apenas usado nas funções de escrita. O intervalo deste valor depende do objeto em causa. Pode ser expresso em decimal ou hexadecimal (adicionando o prefixo '0x').

No seguinte exemplo podemos ver a consulta da posição atual(objecto 0x6063) do controlador com ID 2[16].

```
0x02 R 0x6063                // Sent message by host
0x02 W 0x6063 0x2130         // Received message from controller
```

No seguinte exemplo podemos ver a atribuição do valor 2000 à posição imposta (objeto 0x607A) do controlador com ID 2[16].

```
2 W 0x607A 2000              // Sent message by host
```

No seguinte exemplo podemos ver a atribuição do valor 100000 ao objecto 0x2500 com o SubIndex 7 do controlador 12[16].

```
12 W 0x72500 100000          // Sent message by host
```

O valor do comando enviado pelo controlador pode ser expresso em formato decimal ou hexadecimal. Para configurar o formato usa-se o objeto 0x2000, SubIndex 4[16].

3.1.2 Maquina de estados

Os vários estados de um controlador emcl são mostrados na Figura 3.10[16]:

O estado do controlador pode ser modificado recorrendo ao objeto 0x6040 (Controlword) e pode ser consultado com o objeto 0x6041 (Statusword). Algumas mudanças de estado dão-se de forma automática. O Statusword também permite obter informação sobre condições do sistema, por exemplo, ao impormos uma velocidade, quando essa velocidade é alcançada é ativada uma flag no Statusword[16].

A Figura 3.12 mostra os 16 bits do objeto Controlword cujas combinações geram comandos que mudam o estado do controlador. A Figura 3.13 mostra os comandos possíveis de gerar[16].

Os campos marcados com o 'X' são irrelevantes e os campos não mostrados são constantes ou dependem o mode de operação como é o caso dos campos 'Mode specific'[16].

A Figura 3.15 mostra o respetivo estado do controlador consoante a interpretação dos 16 bits do objeto 'Statusword'[16].

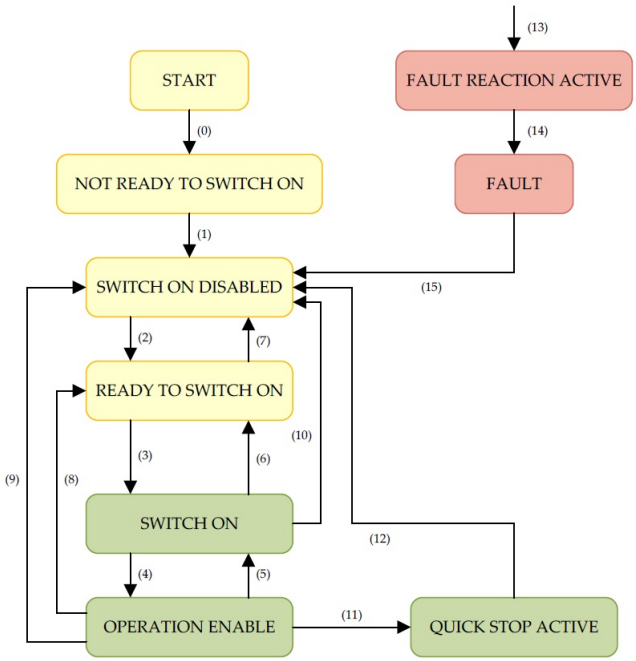

Figura 3.10: Os vários estados de um controlador emcl[16].

Controlword

Index	Sub Index	Name	Data Type	Acc.	Pdo Map.	NVM	Value range	Default value	Units
0x6040	0x00	Controlword	UINT16	RW	Yes	No	UINT16	0x0000	-

Figura 3.11: O objeto Controlword[16]

3.1.3 Modos de operação

O controlador possui os seguintes modos de operações[16]:

- Open loop scalar mode: Este modo permite o excitamento do motor em malha aberta, controlando apenas a tensão aplicada.
- Homing mode: Este modo proporciona vários métodos para localizar a posição inicial pretendida.
- Profile position mode: Este modo permite o posicionamento do sistema. A velocidade e a aceleração podem ser controladas.
- Profile velocity mode: Este modo permite impor uma determinada velocidade ao sistema.
- Profile torque mode: Este modo permite impor um determinado torque ao sistema
- Cyclic sync position mode: Este modo permite o posicionamento do sistema.

Tabela 3.1: Transições entre os vários estados[16].

Trasição	**Evento**	**Ação**
0	Transição automática após ligar o controlador ou após ser feito o reset	É feito um teste diagnostico e o dispositivo é inicializado
1	Transição automática após inicialização	As comunicações são ativadas
2	Trasição ocorre quando o comando "Shutdowndown" é executado	Nenhuma
3	Trasição ocorre quando o comando "Switch on" é executado	A potência de alto nível é ativada
4	Trasição ocorre quando o comando "Enable operation" é executado	A função drive é ativada, a etapa phasing é executada e todos os set-points internos são zerados
5	Trasição ocorre quando o comando "Disable operation" é executado	A função driver é desativada
6	Trasição ocorre quando o comando "Shutdowndown" é executado	A potência de alto nível é desativada
7	Trasição ocorre quando os comandos "Quick stop" ou "disable voltage" são executados	Nenhuma
8	Trasição ocorre quando o comando "Shutdowndown" é executado	A função driver é desativada assim como a potência de alto nível
9	Trasição ocorre quando o comando "Disable operation" é executado	A função driver é desativada assim como a potência de alto nível
10	Trasição ocorre quando os comandos "Quick stop" ou "disable voltage" são executados	A potência de alto nível é desativada
11	Trasição ocorre quando o comando "Quick stop" é executado	A função "quick stop" é inicializada
12	Transição ocorre quando a função "quick stop" termina ou quando o comando "disable voltage" é executado	A função driver é desativada assim como a potência de alto nível
13	Transição ocorre quando é recebido o sinal "fault"	A função configurada para responder a uma quotesfault é executada
14	Transição automática	A função driver é desativada assim como a potência de alto nível
15	Trasição ocorre quando o comando "Fault reset" é executado	É feito um reset à condição de "fault"

3.1.4 Timers

O controlador incorpora alguns contadores em tempo real disponíveis para o utilizador. Há dois tipos de timers: timers de contagem crescente e timers de contagem decrescente. No caso de um timer crescente, o valor do timer é incrementado a cada 1 ms até chegar ao seu limite máximo e aí é posto novamente a zero. No caso de um timer decrescente, é decrementado a cada 1 ms e quando chega ao seu limite mínimo é novamente colocado no seu máximo. Os seguintes objetos permitem ler e escrever nos timers do controlador[16].

15	...	9	8	7	6	5	4	3	2	1	0
Reserved			*Halt*	*Fault reset*	*Mode specific*			*Enable operation*	*Quick stop*	*Enable voltage*	*Switch on*

Figura 3.12: Formato do valor do objeto Controlword[16].

Command	Bit of the *controlword*				
	Fault reset	Enable operation	Quick stop	Enable voltage	Switch on
Shutdown	0	X	1	1	0
Switch on	0	0	1	1	1
Disable voltage	0	x	x	0	x
Quick stop	0	x	0	1	x
Disable operation	0	0	1	1	1
Enable operation	0	1	1	1	1
Fault reset	0 to 1 (rising edge)	x	x	x	x

Figura 3.13: Comandos possíveis[16]

3.1.5 Interrupts

O emcl implementa um "Macro Interrupt System" que permite versatilidade adicional na programação do controlador de movimento. O sistema possui 10 fontes de interrupção com os vetores correspondentes. Quando uma macro se encontra em execução e uma fonte de interrupção é acionada, essa macro é guardada no stack das macros e a execução da macro definida pelo respetivo vetor de interrupção é iniciada. Este procedimento é idêntico quando é executado o comando Macro Call[16].

A tabela de vetores de interrupção consiste em uma entrada para cada fonte de interrupção. Cada entrada nesta tabela deve ser preenchida com um número de uma macro válida. Se uma interrupção é gerada e a respetiva entrada na tabela não tiver sido definida(valor igual a 0) a interrupção é simplesmente ignorada. Isto implica que a macro 0 não possa ser usada como macro ligada a uma interrupção. Na Figura 3.19 são apresentadas todas as fontes de interrupção[16].

3.1.6 Raspberry Pi Model 3B

Principais caracteristicas:

- CPU: 4 x Cortex-A53 1.2 GHz.
- Conjunto de instruções: ARMv8-A 64-bit.
- Memoria: 1 GB (SDRAM) partilhada com a Gpu.
- Conectividade: 4 x USB 2.0, HDMI (rev 1.3), CSI, DSI.
- Rede: 10/100 Mbit/s Ethernet, 802.11n Wireless, Bluetooth 4.1

Statusword

Index	Sub Index	Name	Data Type	Acc.	Pdo Map.	NVM	Value range	Default value	Units
0x6041	0x00	Statusword	UINT16	RO	Yes	No	UINT16	-	-

Figura 3.14: O objeto Statusword[16].

Value (binary)	Status
xxxx xxxx x0xx 0000	Not ready to switch on
xxxx xxxx x1xx 0000	Switch on disabled
xxxx xxxx x01x 0001	Ready to switch on
xxxx xxxx x01x 0011	Switched on
xxxx xxxx x01x 0111	Operation enabled
xxxx xxxx x00x 0111	Quick stop active
xxxx xxxx x0xx 1111	Fault reaction active
xxxx xxxx x0xx 1000	Fault

Figura 3.15: Estados baseados na leitura do Statusword[16].

- Alimentação: 5V por micro USB
- Consumo: 300 mA (1.5 W) média quando em standby, 1.34 A (6.7 W) de consumo máximo.
- Dimensões: 85.60 mm x 56.5 mm x 21 mm

O Raspberry Pi Model 3B possui um processador Cortex-A53 , baseado na arquitetura ARMv8-A 64-bit sendo esta uma arquitetura RISC. Uma arquitetura RISC (do inglês reduced instruction set compute) é caracterizada por um conjunto de instruções pequeno, na ordem das 100 ou menos instruções, tendo estas instruções um baixo tempo de execução. Normalmente as instruções têm um comprimento fixo, tipicamente 4 bytes. As operações aritméticas e lógicas são executadas tendo como operandos apenas registos, sendo as instruções "load" e "store" as únicas que referenciam a memória. Também tem como característica o uso intensivo dos registos, havendo procedimentos que conseguem evitar qualquer referência à memória. Tipicamente esta arquitetura possui um elevado número de registos. Todas estas características fazem com que a arquitetura RISC tenha uma construção mais simples do que a arquitetura CISC que dominada o mercado dos computadores pessoais. Esta construção mais simples permite um menor consumo energético e menor preço de produção o que faz com que a arquitetura RISC tenha tido grande sucesso em sistemas embarcados[17].

De seguida são ilustradas de forma simplificada as conexões entre o Raspberry Pi e os diferentes circuitos integrados que permitem a leitura da força, da posição e a comunicação entre o Raspeberry Pi e o controlador LCC-11. Os esquemas elétricos usados no projeto podem ser consultados na secção "Desenvolvimento das placas pcb".

Modes of operation

Index	Sub Index	Name	Data Type	Acc.	Pdo Map.	NVM	Value range	Default value	Units
0x6060	0x00	Mode of operation	INT8	RW	Yes	Yes	INT8	1	-

Figura 3.16: Objeto Modes of operation[16].

Value	Modes of operation
-1	*Open loop scalar mode*
1	*Profile position mode*
3	*Profile velocity mode*
4	*Profile torque mode*
6	*Homing mode*
8	*Cyclic sync position mode*

Figura 3.17: Os vários modos de operação[16].

3.1.7 Leitura da Força

A leitura da força é feita recorrendo à célula de carga e ao módulo HX711 da Sparkfun ilustrado na Figura 3.21. Este módulo contem o circuito integrado HX711 que é um conversor de analógico para digital com 24-bit de precisão que pode interagir diretamente com um sensor ponte usado para medir cargas. O multiplexer de entrada permite escolher entre o Canal A e o Canal B como input para o PGA (programmable gain amplifier). O canal A pode ser programado com um ganho de 128 ou 64. O Canal B tem um ganho fixo de 32. O regulador da fonte de alimentação interno elimina a necessidade de um regulador externo que forneça potência analógica para o ADC e para o sensor. O input do clock é flexível. Ele pode ter origem num clock externo, um cristal ou um oscilador interno que não requer nenhum componente externo. Não há a necessidade de programar os registos internos pois todo o controlo é feito através dos pinos.

3.1.8 Leitura da Posição

O circuito integrado LS7366R é um contador de 32-bit que permite a leitura de clocks de quadratura de encoders incrementais. Para comunicações com micro-controladores e microprocessadores o integrado possui barramento SPI/MICROWARE com 4 fios. Os 4 fios são SS, SCK, MISO e MOSI. A transferência de dados entre o micro-controlador e o slave LS7366R é síncrona. A sincronização é feita pelo clock SCK fornecido pelo micro-controlador. Cada transmissão de dados é realizada em blocos de 1 a 5 bytes. O ciclo da transmissão é iniciado com uma transição do input SS de HIGH para LOW. O primeiro byte recebido num ciclo de transmissão é sempre um byte de instrução enquanto os bytes do segundo ao quinto são sempre interpretados como dados. A transmissão termina com uma transição de low para high do input SS. Os bytes são recebidos no input MOSI, MSB em primeiro com as transições de low para high do clock SCK enquanto os bytes são enviados pelo output MISO, MSB em primeiro com as transições de high para low do clock SCK.

Timer access

Index	Sub Index	Name	Data Type	Acc.	Pdo Map.	NVM	Value range	Default value	Units
0x2C06	0x01	Timer 1 (count up) value	UINT32	RW	No	No	UINT32	0	milliseconds
0x2C06	0x02	Timer 2 (count up) value	UINT32	RW	No	No	UINT32	0	milliseconds
0x2C06	0x03	Timer 3 (count down) value	UINT32	RW	No	No	UINT32	0	milliseconds
0x2C06	0x03	Timer 4 (count down) value	UINT32	RW	No	No	UINT32	0	milliseconds

Figura 3.18: Os vários objetos Timers[16].

Interrupt vector	Interrupt source	Interrupt vector	Interrupt source
10	Reserved	5	Timer 1 overflow
9	User i²t exceeded	4	GPI4 state enabled
8	Timer 4 underflow	3	GPI3 state enabled
7	Timer 3 underflow	2	GPI2 state enabled
6	Timer 2 overflow	1	GPI1 state enabled

Figura 3.19: Lista das fontes de interrupção[16].

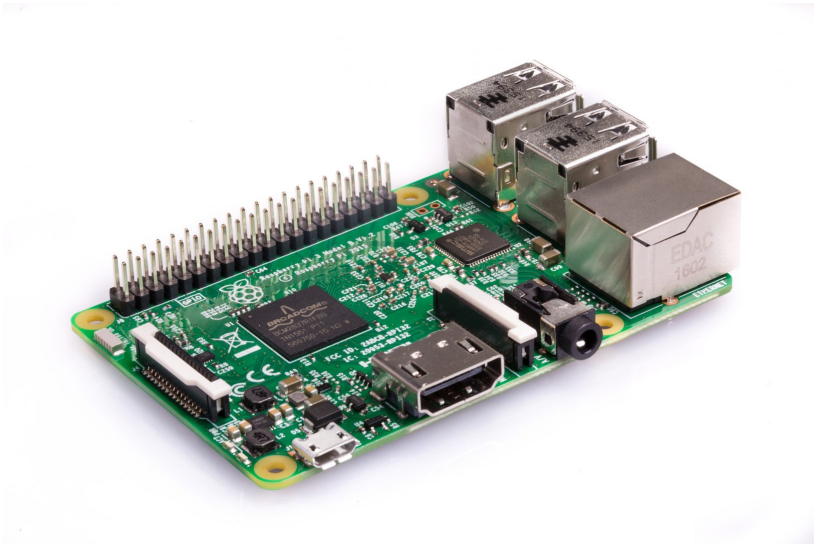

Figura 3.20: O Raspberry Pi 3 B.

3.1.9 Comunicação entre Raspeberry Pi e LCC-11

A comunicação é feita por RS232. Como o Raspeberry Pi utiliza tensões TTL/CMOS é necessário uma conversor de níveis de tensão. Neste projeto foi usado o MAX232N da Texas Instrument, que permite a conversão entre níveis TIA/EIA-232-F e TTL/CMOS. Na Figura 3.27 é ilustrada de maneira simplificada as conexões.

Figura 3.21: Módulo HX711 para a leitura da carga.

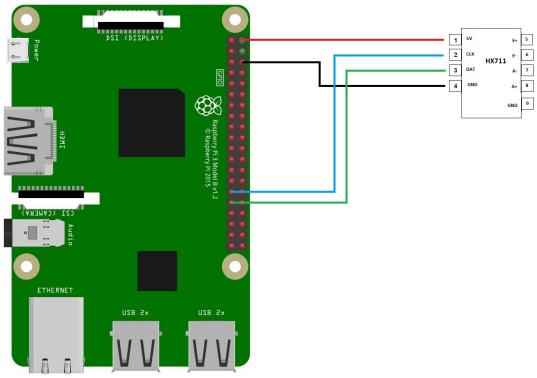

Figura 3.22: Esquema simplificado das ligações ao módulo HX711.

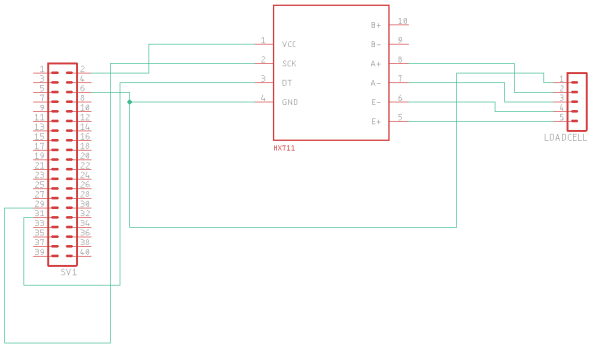

Figura 3.23: Esquema elétrico das ligações ao modulo HX711.

Figura 3.24: O circuito integrado LS7366R.

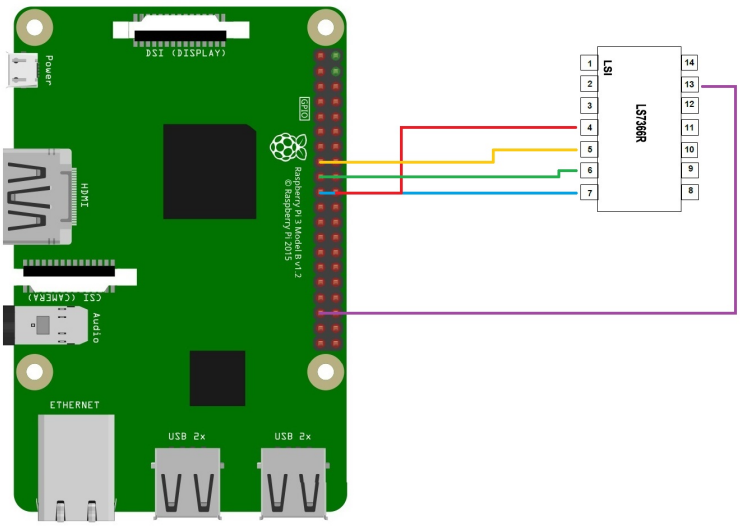

Figura 3.25: Esquema simplificado das ligações ao circuito integrado LS7366R.

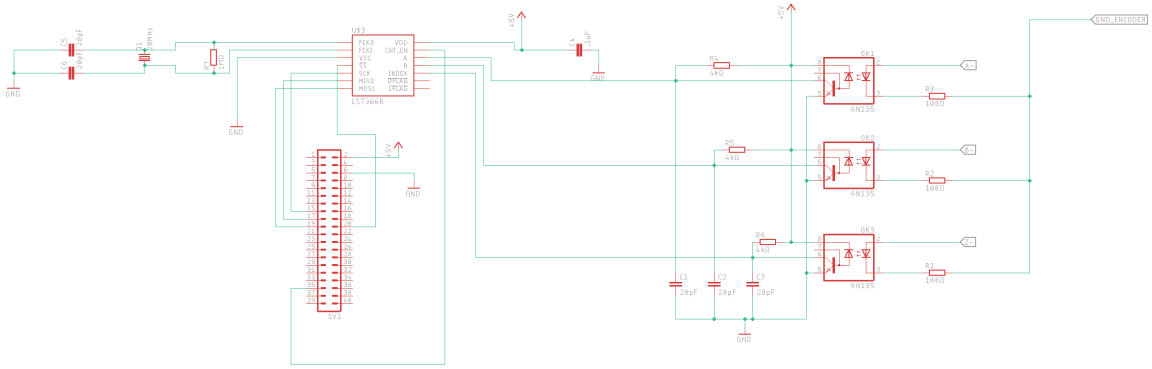

Figura 3.26: Esquema elétrico das ligações ao circuito integrado LS7366R.

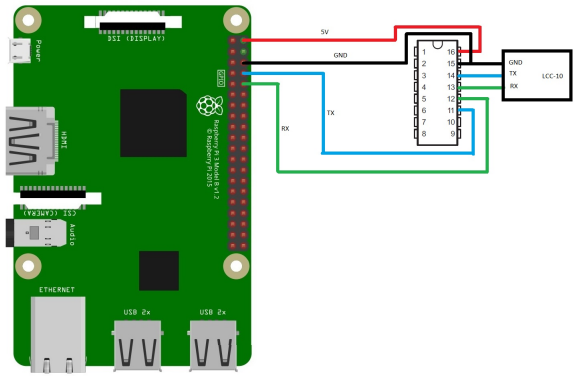

Figura 3.27: Esquema simplificado das ligações ao circuito integrado MAX232N.

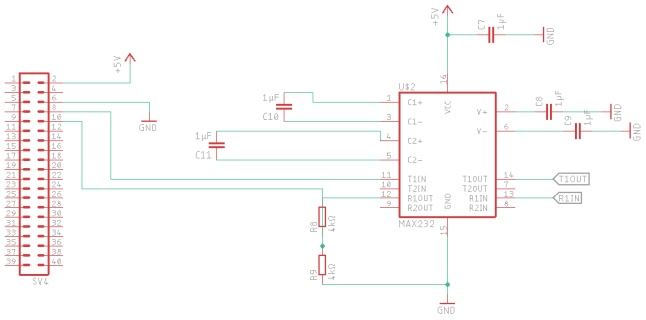

Figura 3.28: Esquema elétrico das ligações ao circuito integrado MAX232N.

3.1.10 Sinais enviados para o LCC-11

O controlador LCC-11 possui 4 entradas digitais com uma resistência de pull-up de 4K7. Neste projeto o input 1 é usado para assinalar que o ensaio a decorrer terminou enquanto o input 2 é usado para indicar o limite de um ensaio cíclico ou seja quando há uma transição de estado deste input o controlador LCC-11 inverte o sentido do movimento sendo que estes sinais são gerados pelo Raspberry Pi. Para este fim, são usados 2 opto-isoladores de transístor. Este opto-isolador é ilustrado na 3.29. Ao fazer passar uma determinada corrente no elemento transmissor, o transístor recetor abre e o input do controlador LCC-11 é posto à tensão do ponto 3 que neste caso está ligado ao GND. Com o transístor recetor ao corte o input tem uma tensão de 5V devido à resistência interna de pull-up.

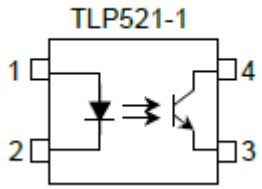

Figura 3.29: Opto-isolador de transistor.

3.2 Sistema Operativo

O sistema operativo usado neste projeto foi o Raspbian que é baseado na distribuição Debian do Linux. O Raspbian é o sistema operativo oficial da Raspberry Pi Foundation e é otimizado para os processadores ARM que constituem os vários modelos Raspeberry Pi. Para além disso este sistema operativo contém uma série de bibliotecas pré-instaladas para tirar partido das funcionalidades I/O do Raspberry Pi.

3.3 Software desenvolvido

Neste projeto foi desenvolvido um programa local em C que corre no Raspberry Pi e um programa remoto em HTML/Javascript que corre no browser. O programa principal permite fazer todo o controlo do controlador e comunica com uma base de dados instalada no Raspberry Pi. Este programa interage com o utilizador através de uma interface gráfica feita em GTK. A base de dados guarda o resultados dos vários ensaios efetuados assim como definições do utilizador.

Em computação, um processo é uma abstração do sistema operativo de um programa em execução. Vários processos podem executar concorrentemente no mesmo cpu e cada processo é dada a ilusão ter o uso exclusivo do hardware. Concorrência significa que as instruções de vários processos são intervaladas no tempo. No caso de existir mais que um processador, os processos são executados em paralelo ou seja são executados simultaneamente nos vários processadores. Em ambos os casos um único cpu executa processos concorrentemente mudando rapidamente entre eles. O sistema operativo alterna entre os vários processos executando o "context switch".

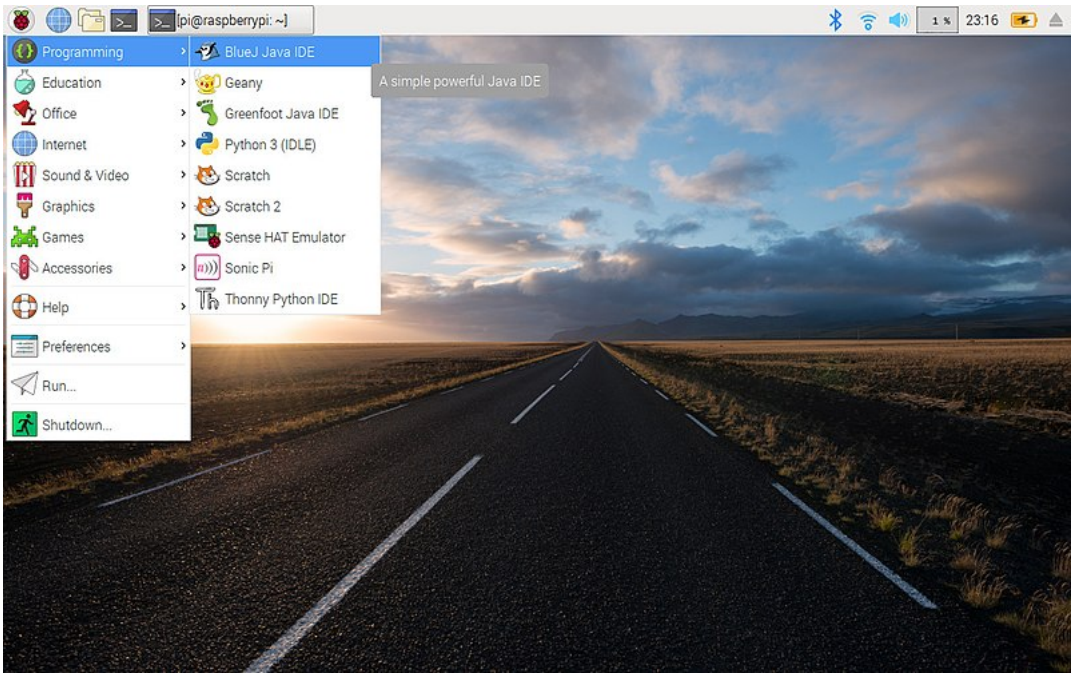

Figura 3.30: O sistema operativo Raspbian.

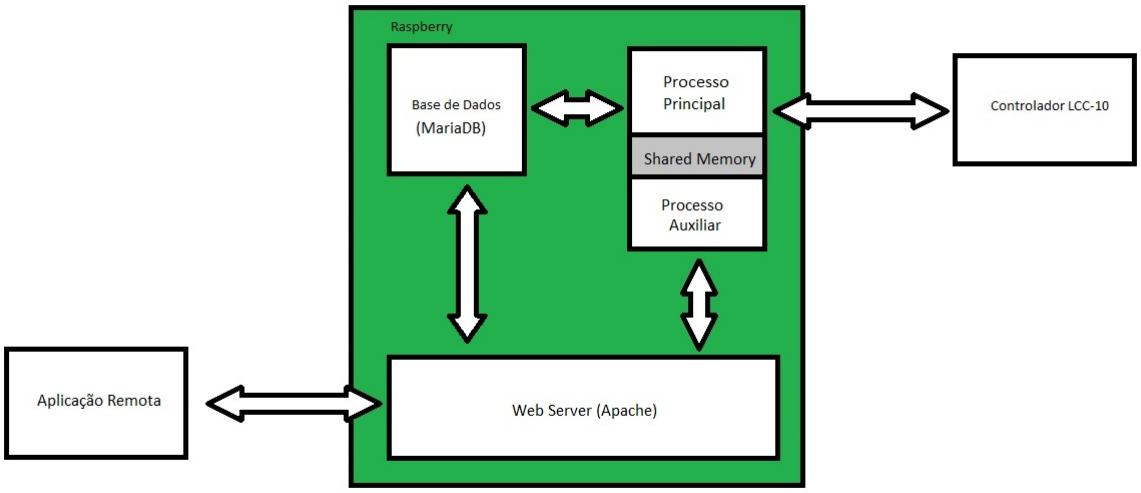

Figura 3.31: Esquema do hardware e software usado.

No início da computação um processo era constituído por um único "control flow". Nos sistemas modernos podemos ter um processos com vários "control flow" sendo cada um deles denominado por uma thread. Todas as threads pertencentes a um processo correm no seu contexto e partilham o código e os dados globais. As threads são um modelo de programação com uma cada vez maior importância, uma vez que é mais fácil partilhar dados entre threads do que entre processos e porque as threads são normalmente mais eficientes do que os processos. Multi-threading resulta à partida na execução mais rápida do programa quando este corre num sistema com mais que um processador. Internamente o Linux trata todas as entidades de execução como threads sendo que um processo mais não é que um conjunto de 1 ou mais threads que partilham o "address space" entre outros elementos.

3.4 Aplicação Local

3.4.1 Arquitetura da aplicação

Neste projeto, o programa local é constituído por 3 threads principais. A thread inicial (thread 1) é a thread responsável pela interface gráfica (gtk). Esta thread gere também toda a comunicação UART com o controlador. A thread 1 cria a thread 2 e a thread 3. A thread 2 é responsável pela leitura da posição recorrendo ao protocolo spi e pela leitura da força recorrendo ao protocolo i2c. Esta leitura é realizada em intervalos de tempo pré-determinados sendo por defeito 100ms. A thread 3 tem como função fazer polling na "Shared Memory" esperando por comandos caso o programa se encontre em modo de controlo remoto. Para além destas 3 threads principais, a thread 1 também cria automaticamente outras threads auxiliares referentes à biblioteca gtk. A thread 2 e a thread 3 correm no core 3 e 4 respetivamente e executam com prioridade real-time. Embora o sistema operativo Linux não seja uma sistema operativo real-time, o seu schedule possui 2 modos que tentam aproximar ao máximo um sistema com estas características e são chamados de soft real-time ao contrário de hard real-time. Estes dois modos são SCHED_RR (round-robin) e SCHED_FIFO (first-in, first-out). Para as threads 2 e 3 foi usado o SCHED_FIFO sendo que a grande diferença para o SCHED_RR é que este modo não possui time-slices. Time-slices é um intervalo de tempo a partir do qual a thread que está a executar é suspensa e é dada a oportunidade a outra. Por conseguinte, a thread executa até voluntariamente libertar o cpu, até terminar ou até ser abortada por uma thread com prioridade mais elevada.

3.4.2 Macros

No controlador os comandos podem ser inseridos diretamente e executados imediatamente. O emcl também oferece a capacidade de definir conjuntos de comandos, "macros". Este comandos são guardados na memória não volátil (NVM) e podem ser executados automaticamente. As macros são criados concatenando um comando com um argumento e indicando o local onde ficará guardado. O controlador permite a criação de 64 macros, cada uma contendo 64 comandos. Para além disso também estão disponíveis alguns registos, assim como comandos aritméticos de sequência que permite alguma capacidade de programar em "high level". Todas as macros devem ser terminadas com a instrução "Return from macro call". A Macro '0' é executada automaticamente quando o controlador

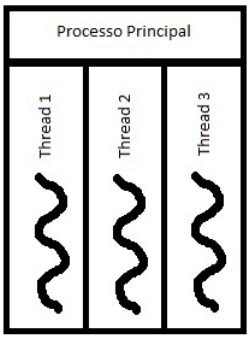

Figura 3.32: As threads da aplicação local.

é ligado[16].

Macro access

Index	Sub Index	Name	Data Type	Acc.	Pdo Map.	NVM	Value range	Default value
0x2C05	0x01	Macro number	UINT8	RW	No	No	UINT8	-
0x2C05	0x02	Macro command	UINT8	RW	No	No	UINT8	-
0x2C05	0x03	Command	UINT64	WO	No	No	UINT64	-
0x2C05	0x04	Protected access	UINT8	RW	No	Yes	0-1	0

Figura 3.33: Os vários objetos referentes à programação das macros[16].

3.4.3 Fase Inicial

Ao ser iniciado o programa, é ativado uma saída digital do raspberry pi que ativa um relé que por sua vez alimenta o controlador. Sempre que inicia, o controlador executa a macro 0. Neste projeto a macro 0 foi programada para executar o homing do sistema e colocar o controlador no modo de operação velocidade. O homing pode ser feito em relação ao limite mecânico negativo ou positivo ou ao primeiro index que o controlador encontra depois de passar pelo limite mecânico negativo ou positivo. De seguida são apresentados de forma detalhada os passos quando o controlador é iniciado.

1. O Controlador é iniciado
2. Começa a etapa phasing
3. É invocada a macro 0
4. A macro 0 entra em loop até a etapa phasing estar concluída
5. A macro 0 muda o modo de operação para homing e inicia a etapa homing que é realizada em relação ao primeiro index após o limite mecânico negativo.
6. A macro 0 entra em loop até a etapa homing estar concluída.

7. A macro 0 muda o modo de operação para velocidade

8. O controlador está pronto a ser usado

3.4.4 Menu Machine

Na interface gráfica, o primeiro menu é o menu Machine, ilustrado na Figura 3.34 que permite escolher de entre vários controladores guardados, adicionar um novo controlador ou eliminar um previamente guardado. A mesma coisa se aplica às células de carga. As respetivas imagens são guardadas localmente na pasta /usr/html para poderem ser acedidas também pela aplicação remota. Estes ficheiros de imagem são copiados no caso da aplicação local e é feito o seu upload no caso da aplicação remota. O mesmo se aplica às células de carga. Toda esta informação é guardada na base de dados. Cada vez que se entra no menu Machine, o programa lê a base de dados e atualiza os respetivos menus de seleção. Quando se seleciona um controlador ou uma célula de carga, esta seleção também é guardada na base de dados de modo a manter-se na próxima vez que se iniciar o programa.

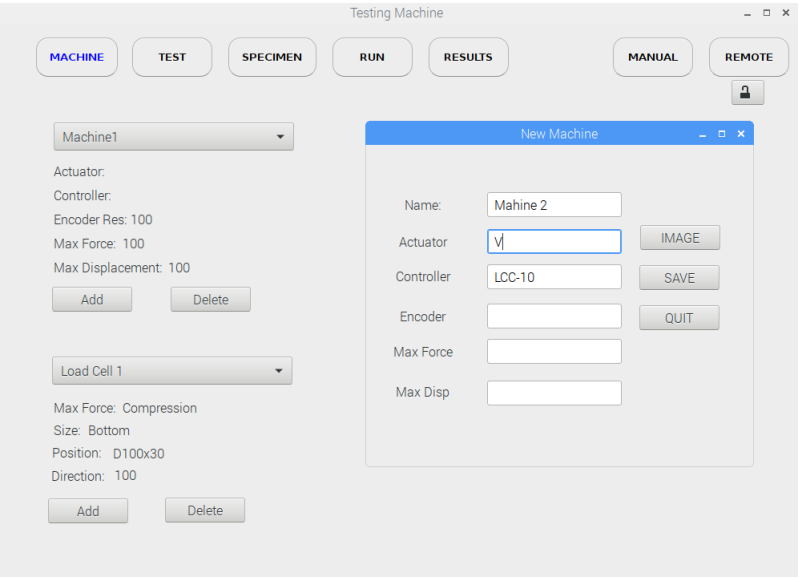

Figura 3.34: Menu Machine.

3.4.5 Menu Test

O menu Test, ilustrado na Figura 3.35, permite escolher o tipo de teste a efetuar. Existem 6 tipos de testes: Tension, Compression, 3 point Bending, 4 Point Bending, Cyclic Force e Cyclic Displacement. Estes testes são programados em macros. Embora fisicamente diferentes, os 4 primeiros testes são iguais ao nível da programação e por isso usam a mesma macro. A única diferença prende-se com a configuração dos ensaios. Neste menu também é possível definir presets de testes. Estes presets são guardados na base de dados. Quando se entra no menu o programa lê a base de dados e compõe o menu de seleção. Os presets são apresentados por ordem alfabética e é possível eliminar presets

anteriormente criados. Ao selecionar um presets os campos respetivos são preenchidos deixando ao utilizador a opção de manter ou alterar estes valores.

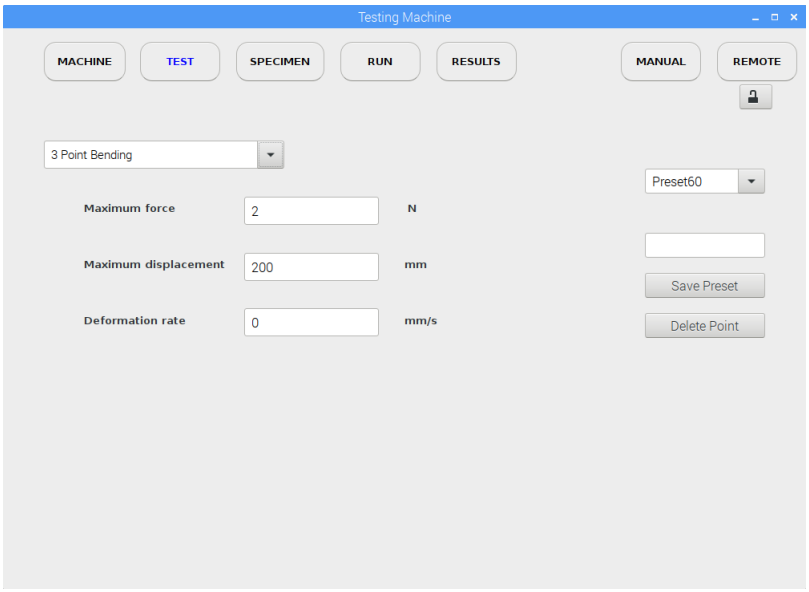

Figura 3.35: Menu Test.

Os testes Tension, Compression, 3 point Bending e 4 Point Bending utilizam a Macro 1. O teste Cyclic Force utiliza a macro 2 e o teste Cyclic Dysplacement utiliza a macro 3. Para além destas são usadas outras macros abordadas mais à frente. A Figura 3.36 mostra a disposição das macros na memoria do controlador. Todas as macros são guardadas de forma não volátil no controlador e podem sempre ser reprogramadas no menu Manual->Setup.

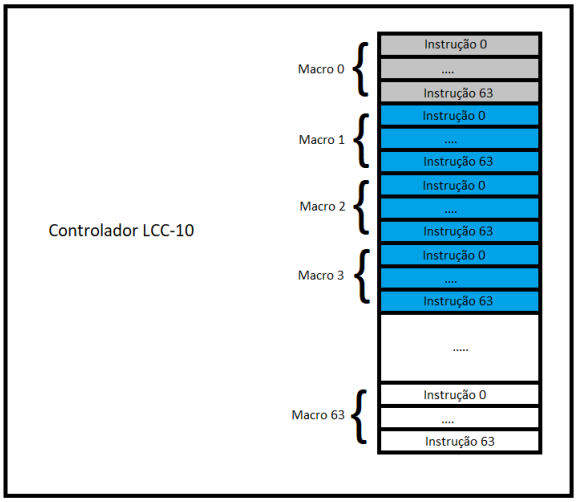

Figura 3.36: Mapa das macros no controlador.

3.4.6 Menu Specimen

O menu Specimen, ilustrado na Figura 3.37, permite escolher a geometria da amostra a ser testada. Esta geometria pode ser T-Bone, Rectangular, Wire, Block, Cylinder, Beam 3-point e Beam 4-point. Dentro de cada geometria o utilizador pode escolher os vários parâmetros que a definem. Estes parâmetros são usados mais tarde nos cálculos necessários e nas restrições que se possam aplicar devido a essa geometria.

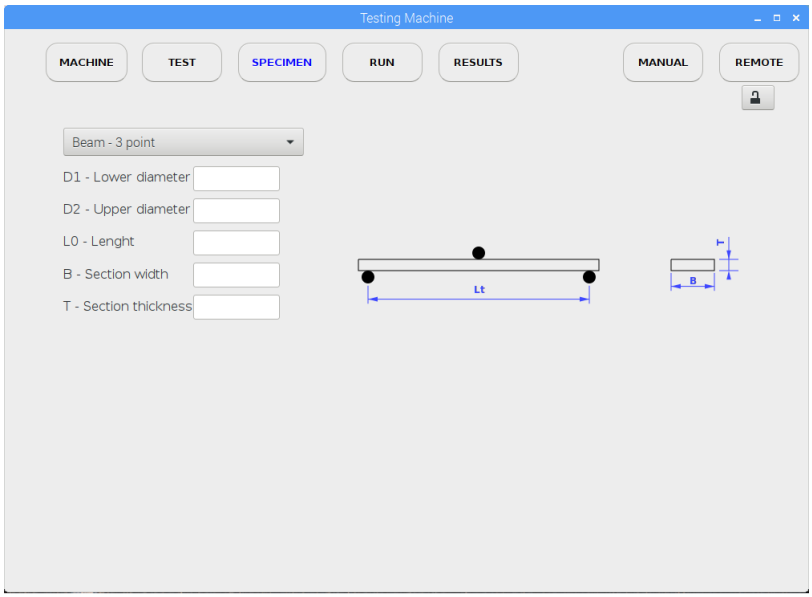

Figura 3.37: Menu Specimen.

3.4.7 Menu Run

O menu "Run", ilustrado na Figura 3.39, é o menu onde é possível dar inicio aos ensaios. Ao carregar no botão "Iniciar Ensaio" é iniciado o ensaio escolhido previamente no menu "Test". O ensaio decorre até ser atingida uma condição de paragem. A qualquer momento o ensaio também pode ser interrompido ao carregar no botão "Parar". Tanto no caso de ser atingida uma condição de paragem ou no caso de ter sido carregado o botão "Parar", o programa coloca o pin 26 em "HIGH" o que permite à macro saber que o ensaios tem de ser terminado.

No Caso de ter sido escolhido os teste "Tension", "Compression", "Beam 3-point" ou "Beam 4-point", são executados os passos a seguir descritos, e todos os comandos trocados entre o Raspberry Pi e o Controlador para este tipo de ensaio são mostrados na Tabela 3.2.

Como mostrado nas tabelas seguintes, quando se pretende inserir uma instrução numa macro, usa-se o comando "0x20 W 0x32C05 x" em que o x é número resultante da representação binária do comando que vai ocupar aquela posição na macro. Este número é composto por 8 bytes cada um com o seu significado como mostra a Figura 3.38.

1. O programa envia uma mensagem por RS232 para o controlador para ler a posição atual do controlador e obtém a resposta .

7	6	5	4			3	2	1	0
SubIndex (8bits)	Index (16bits)		Length (4bits)	Reserved (3bits)	R/W (1bit)	Data (32bits)			

Figura 3.38: Formato do comando a ser inserido numa dada posição numa macro.

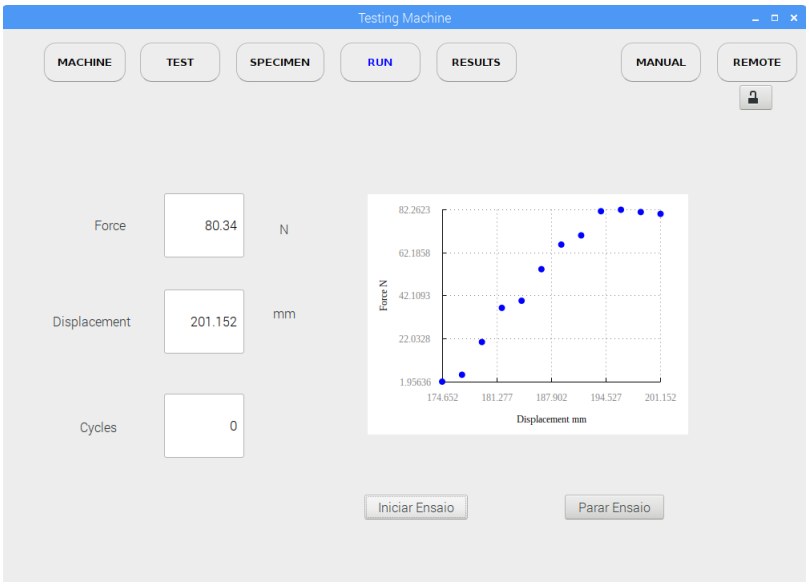

Figura 3.39: Menu Run.

2. A partir do limite de posição escolhido pelo utilizador, o programa calcula o limite de posição do controlador no qual o ensaio irá decorrer.

3. O programa envia uma mensagem por RS232 para substituir na macro 1 a respetiva instrução que define a velocidade de deformação.

4. O programa envia a mensagem por RS232 para o controlador iniciar a execução da macro 1 e o ensaio é iniciado.

5. A thread 2, responsável pela leitura da força e da posição, após serem efetuadas estas leituras, compara os valores lidos com os valores limite e coloca o pin 26 em "HIGH" caso se verifiquem as condições de paragem.

6. A macro, que constantemente monitoriza o pin 26, termina o ensaios quando este é colocado a "HIGH".

7. No decorrer do ensaio é apresentado um gráfico no menu "Run" que vai sendo atualizado com os valores lidos da posição e da força.

8. Quando termina o ensaio os valores da posição e da força previamente guardados na memória alocada dinamicamente são transferidos para a base de dados assim como o instante em que foram lidos. O nome do ensaios é obtido através da concatenação da data, da hora, minutos e segundos de modo a ser um identificador único.

Tabela 3.2: Comandos trocados entre o Raspberry Pi e o Controlador LCC-11 nos 4 primeiros tipos de ensaio.

Origem	**Valor**	**Descrição**
R	32 R 0x6064\r	O Raspberry envia um comando a pedir a leitura da posição do controlador
C	32 W 0x6064 10000\r	O Controlador devolve o valor da posição atual, neste caso 10000.
R	32 W 0x12C05 1\r	Seleciona a macro 1
R	32 W 0x22C05 3\r	Seleciona a instrução 3 da macro 1
R	32 W 0x32C05 x\r	Escreve a instrução em que x é a representação binária da instrução que determina a velocidade do ensaio
R	32 W 0x12C04 1\r	Inicia a macro 1

No caso de um ensaio cíclico de força são executados os passos a seguir descritos e todos os comandos trocados entre o Raspberry Pi e o Controlador para este tipo de ensaio são mostrados na Tabela 3.3.

1. O programa envia uma mensagem por RS232 para o controlador e obtém como resposta a posição atual.

2. A partir do limite de posição escolhido, o programa calcula o limite de posição do controlador no qual o ensaio irá decorrer.

3. O programa calcula a velocidade de deformação através dos limites de posição e da frequência do ensaio.

4. O programa envia uma mensagem por RS232 para substituir na macro 2 as respetivas instruções que definem a velocidade de deformação e os limites de posição.

5. O programa envia a mensagem por RS232 para o controlador iniciar a execução da macro 2.

6. A thread 2, responsável pela leitura da força e da posição, após serem efetuadas estas leituras, compara os valores lidos com os valores limite e coloca o pin 26 em "HIGH" caso se verifiquem as condições de paragem.

7. A macro, que constantemente monitoriza o pin 26, termina o ensaios quando este é colocado a "HIGH".

8. Entretanto, enquanto o ensaio não é terminado, a macro monitoriza também o pin 28 e muda o sentido da velocidade quando este transita de "HIGH" para "LOW" (muda para sentido descendente) ou de "LOW" para "HIGH" (muda para sentido ascendente).

9. No decorrer do ensaio é apresentado um gráfico no menu "Run" que vai sendo atualizado com os valores lidos da posição e da força.

10. Quando termina o ensaio os valores da posição e da força previamente guardados no memória alocada dinamicamente são transferidos para a base de dados assim como o instante em que foram lidos. O nome do ensaios é obtido através da concatenação da data, da hora, minutos e segundos de modo a ser um identificador único.

Tabela 3.3: Comandos trocados entre o Raspberry Pi e o Controlador LCC-11 num ensaios cíclico de força.

Origem	**Valor**	**Descrição**
R	32 R 0x6064\r	O Raspberry envia um comando a pedir a leitura da posição do controlador
C	32 W 0x6064 10000\r	O Controlador devolve o valor da posição atual, neste caso 10000.
R	32 W 0x12C05 1\r	Seleciona a macro 2
R	32 W 0x22C05 9\r	Seleciona a instrução 9 da macro 2
R	32 W 0x32C05 x\r	Escreve a instrução em que x é a representação binária da instrução que determina a velocidade da fase descendente tendo em conta a frequência
R	32 W 0x22C05 14\r	Seleciona a instrução 14 da macro 2
R	32 W 0x32C05 x\r	Escreve a instrução em que x é a representação binária da instrução que determina a velocidade da fase ascendente tendo em conta a frequência
R	32 W 0x12C04 2\r	Inicia a macro 2

No caso de um ensaio cíclico de deslocamento são executados os passos a seguir descritos e todos os comandos trocados entre o Raspberry Pi e o Controlador para este tipo de ensaio são mostrados na Tabela 3.4.

1. O programa envia uma mensagem por RS232 para o controlador e obtém como resposta a posição atual.

2. O programa determina os limites de posição nos quais o ensaio irá decorrer.

3. O programa envia uma mensagem por RS232 para substituir na macro 3 as respetivas instruções que definem os limites de posição.

4. O programa envia a mensagem por RS232 para o controlador iniciar a execução da macro 3.

5. A thread 2, responsável pela leitura da força e da posição, após serem efetuadas estas leituras, compara os valores lidos com os valores limite e coloca o pin 26 em "HIGH" caso se verifiquem as condições de paragem.

6. A macro, que constantemente monitoriza o pin 26, termina o ensaios quando este é colocado a "HIGH".

7. Entretanto, enquanto o ensaio não é terminado, a macro monitoriza também o pin 28 e muda o sentido da velocidade quando este transita de "HIGH" para "LOW" (muda para sentido descendente) ou de "LOW" para "HIGH" (muda para sentido ascendente).Esta velocidade é calcula tendo em conta a posição em que aconteceu o limite anterior e a posição em que ocorreu o limite atual. Assim a macro sabendo a frequência do ensaios calcula a velocidade para o novo ciclo. Este método tem de ser adotado pois ao contrário do ensaio cíclico de deslocamento não é possível calcular previamente a velocidade do ensaios pois não se conhecem as posições em que ocorreram os limites da força.

8. No decorrer do ensaio é apresentado um gráfico no menu "Run" que vai sendo atualizado com os valores lidos da posição e da força.

9. Quando termina o ensaio os valores da posição e da força são guardados na base de dados assim como o instante em que foram lidos. O nome do ensaios é obtido através da concatenação da data, da hora, minutos e segundos e modo a ser um identificador único.

Tabela 3.4: Comandos trocados entre o Raspberry Pi e o Controlador LCC-11 num ensaios cíclico de deslocamento.

Origem	**Valor**	**Descrição**
R	32 R 0x6064\r	O Raspberry envia um comando a pedir a leitura da posição do controlador
C	32 W 0x6064 10000\r	O Controlador devolve o valor da posição atual, neste caso 10000.
R	32 W 0x12C05 3\r	Seleciona a macro 3
R	32 W 0x22C05 6\r	Seleciona a instrução 6 da macro 3
R	32 W 0x32C05 x\r	Escreve a instrução em que x é a representação binária da instrução que determina a velocidade do ensaio
R	32 W 0x22C05 7\r	Seleciona a instrução 7 da macro 3
R	32 W 0x32C05 x\r	Escreve a instrução em que x é a representação binária da instrução que determina o limite superior do ensaio
R	32 W 0x22C05 14\r	Seleciona a instrução 14 da macro 3
R	32 W 0x32C05 x\r	Escreve a instrução em que x é a representação binária da instrução que determina o limite inferior do ensaio
R	32 W 0x12C04 3\r	Inicia a macro 3

O ensaio também pode ser parado a qualquer momento carregando no botão "Parar ensaio". No decorrer do ensaio vão sendo mostrados neste menu os valores da posição, força e número de ciclos atuais.

3.4.8 Menu Results

O menu "Resultados", ilustrado na Figura 3.40, permite a amostragem em gráfico dos ensaios realizados anteriormente. Ao entrar no menu, o programa lê da base de dados o nome dos ensaios guardados e compõe o menu de seleção. Ao ser selecionado um ensaio neste menu, o programa lê novamente da base de dados e obtém todos os pontos da posição e da força que usa para compor o gráfico. O gráfico permite o zoom, usando o ponto onde se carrega e o ponto onde se solta o rato para definir a janela de zoom. Ao carregar no auto-scale o gráfico volta à janela normal. O botão "SAVE GRAPH" permite guardar a imagem do gráfico em formato .png. O utilizador pode escolher a localização e o nome a dar ao ficheiro. Também é possível guardar em ficheiro de texto todos os pontos do gráfico. Cada ponto é guardado numa linha no formato "tempo,posição,força". O gráfico também permite selecionar qualquer ponto e as suas coordenadas exatas são mostradas.

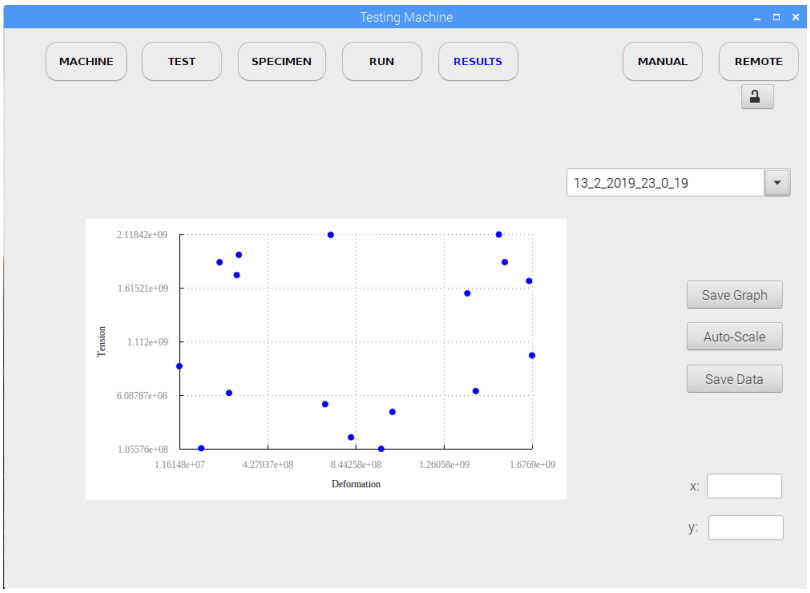

Figura 3.40: Menu Results.

3.4.9 Menu Manual-Manual

O menu "Manual", ilustrado na Figura 3.41, subdivide-se nos sub-menus "Manual" e "Calibration" e "Setup". O submenu "Manual" permite fazer o ajuste manual do controlador. Existe a opção de uma velocidade de 0.5mm/s carregando na figura azul e de 10mm/s carregando na figura amarela. Ao ser pressionada uma destas áreas, é enviada uma mensagem por RS232 para o controlador se deslocar com a respetiva velocidade no sentido selecionado. Se o rato deixar de ser pressionado ou mesmo estando pressionado se sair da área da figura o controlador assume a velocidade 0. A cada instante os valores da posição e da força atuais são mostrados. O controlador tem a opção de guardar internamente até 200 posições diferentes. O programa permite atribuir nomes a estes pontos e guarda na base de dados o respetivo conjunto "nome,index". Para gravar uma posição basta introduzir um nome na "entry" e carregar em "SAVE POINT". O programa consulta a base de

dados para determinar um "index" disponível e faz a associação do nome com essa "index". De seguida manda uma mensagem por RS232 para o controlador gravar a posição atual no respetivo "index". No menu de seleção é possível selecionar um dos pontos e ao carregar no botão "GO TO POINT" é enviada uma mensagem para o controlador para ele assumir a posição associado com o index que o programa por sua vez associou ao nome do ponto. Para eliminar o ponto selecionado basta carregar no botão "DELETE POINT". Também é possível fazer com que o controlador se desloque para o ponto selecionado após o inicio do controlador através do botão "INITIAL POSITION". Também é possível fazer o controlador deslocar-se para uma posição especifica preenchendo a "entry" com o valor pretendido, selecionando a opção "abs" ou "rel" e carregando no botão "GO TO". No caso da opção "abs" o valor é interpretado como uma posição absoluta enquanto no caso da opção "rel" o valor é interpretado como relativo em relação à posição atual do controlador.

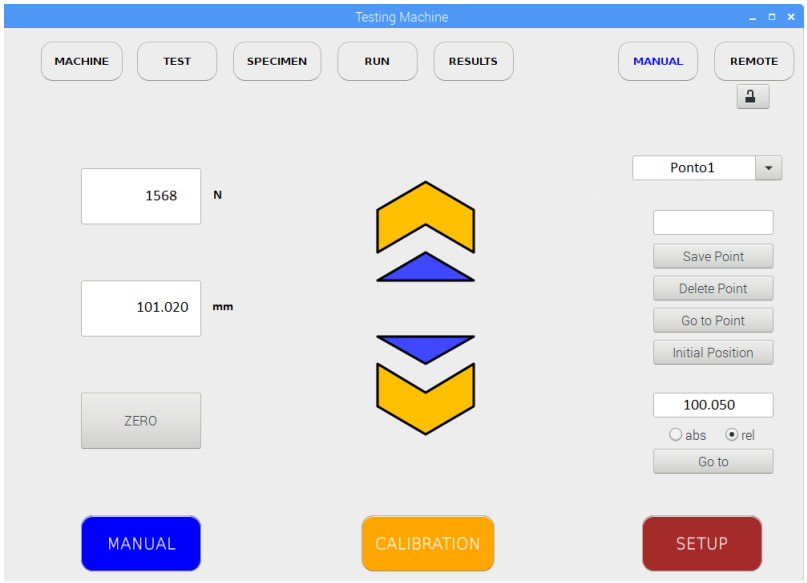

Figura 3.41: Menu Manual->Manual.

3.4.10 Menu Manual-Calibration

No sub-menu "Calibration", ilustrado na Figura 3.44, é possível fazer a calibração da célula de carga. A equação de calibração é do tipo "y=ax+b". A calibração faz-se num primeiro passo onde a célula de carga não tem qualquer peso. Ao carregar-se no "OK" o programa lê 20 valores num espaço de 3 segundos, faz a média e regista o resultado. De seguida coloca-se um peso conhecido, preenche-se a "entry" com o respetivo peso e carrega-se no "OK". O programa faz novamente 20 medições e com esta média e a média com peso zero calcula os valores de "a" e de "b". A partir desse momento usa estes valores para converter o valor lido pelo modulo HX711 no valor da massa real. O valor do "a" e do "b" são guardados na base de dados. A fim de testar a exatidão da medição da força foram efetuados dois ensaios com 20 medições cada. O primeiro ensaio foi efetuado com carga nula e está representado na Figura 3.42. Apresenta um erro máximo de 0,070048 N. O segundo ensaio foi efetuado com uma carga de 6 N e está representado na Figura

3.43. Apresenta um erro máximo de 0,12955 N e um erro relativo de 2,16%.

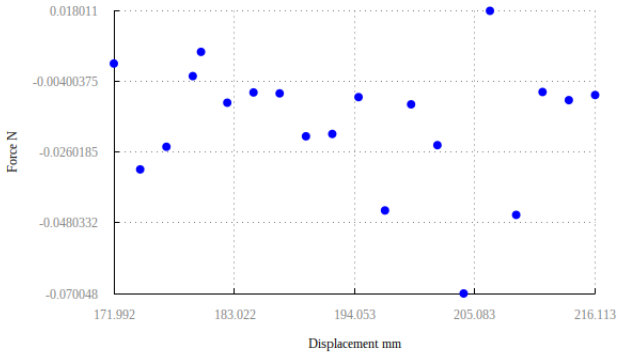

Figura 3.42: Ensaio com carga nula.

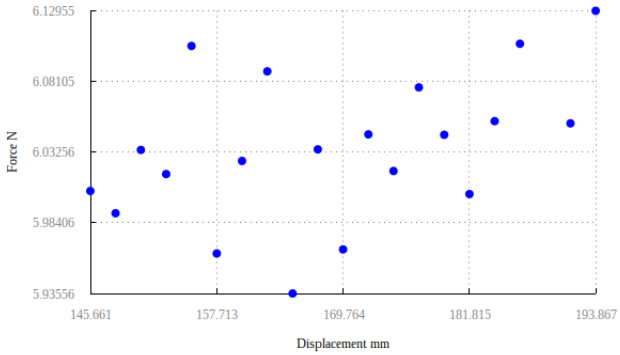

Figura 3.43: Ensaio com carga de 6 N.

3.4.11 Menu Manual-Setup

O sub-menu "Setup", ilustrado na Figura 3.45, é destinado a utilizadores avançados pois permite ler e escrever diretamente nos registos do controlador e destina-se principalmente ao debug do controlador. Na primeira "entry" escreve-se o comando a enviar por RS232 para o controlador. A esta "string" é adicionado o caracter
r. Caso seja um comando de leitura, o valor lido surgirá na "entry 3". Caso seja um comando de escrita, o valor a escrever é obtido da "entry 2" e juntamente com a "string" a "entry 1" é formado o comando de escrita.Este sub-menu também apresenta a opção de reconfigurar a macro 0 carregando no botão "SAVE 0" e reconfigurar todas as macros carregando no botão "SAVE ALL" sendo que as respetivas macros são escritas novamente na memória não volátil do controlador.

Na Tabela 3.5 estão representados os passos para escrever a primeira instrução da macro 1. O segundo e terceiro passos são repetidos para cada instrução da macro. Todo este processo é realizado para todas a macros a guardar no controlador LCC-11.

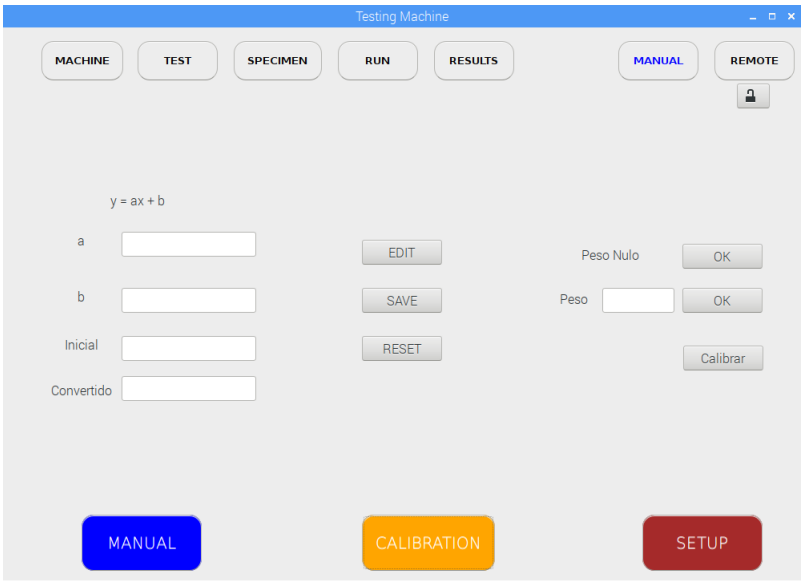

Figura 3.44: Menu Manual->Calibration.

Tabela 3.5: Exemplo da configuração da macro 1.

Origem	**Valor**	**Descrição**
R	32 W 0x12C05 1\r	O Raspberry envia um comando para escrever na macro 1
R	32 W 0x22C05 1\r	O Raspberry envia um comando para escrever na instrução 1
R	32 W 0x32C05 x\r	O Raspberry envia um comando em que x é a representação binária da instrução convertida em decimal

3.4.12 Segurança

O ícone no canto superior direito em forma de aloquete permite ao utilizador bloquear a aplicação. Quando a aplicação é bloqueada todos os botões ficam desativados não sendo possível alterar o estado do programa, servindo sobretudo para usar a aplicação em modo remoto. Para desbloquear a aplicação o utilizador tem de carregar no ícone o na janela que aparece, tem de inserir a password correta. É possível também modificar a password no menu Manual->Setup->Modificar Password. Ao carregar neste botão é aberta uma nova janela onde o utilizador tem de inserir a password antiga e inserir duas vezes a nova password pretendida. A password é encriptada pela Hash Function SHA-256 e o resulto é guardado em forma binária no ficheiro "password" no diretório da aplicação. uma Hash Function é um algoritmo que a partir de um input, gera um resultado de dimensão fixa, neste caso 256 bits. A segurança é fornecida por este algoritmo uma vez que a partir do resultado final é virtualmente impossível deduzir o input, neste caso a password.

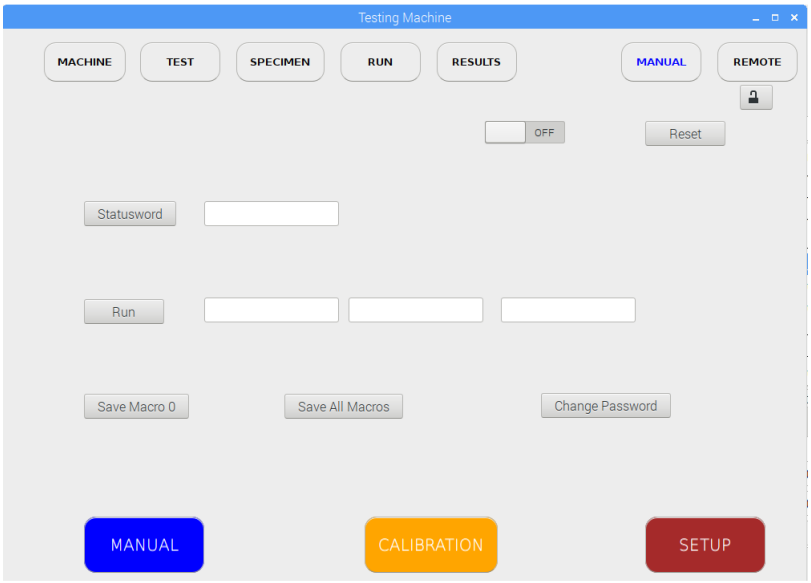

Figura 3.45: Menu Manual->Setup.

3.5 Aplicação Remota

A aplicação remota apresenta todas as funcionalidades que a aplicação local permite. A aplicação web consistem numa única página html e toda a informação que precisa de enviar e receber do servidor é transferida de forma assíncrona pela técnica "Ajax". Esta técnica permite executar um ficheiro .php do lado do servidor que realiza as ações necessárias e caso seja preciso devolve informação que o "Ajax Engine" recebe e por sua vez atualiza a página html. O browser, ao fazer um pedido http para o endereço IP do raspberry pi é servido pelo web server apache com a pagina inicial login.php. Nesta página, o utilizador tem de inserir a password definida pelo programa local para aceder à aplicação web como ilustrada na 3.46.

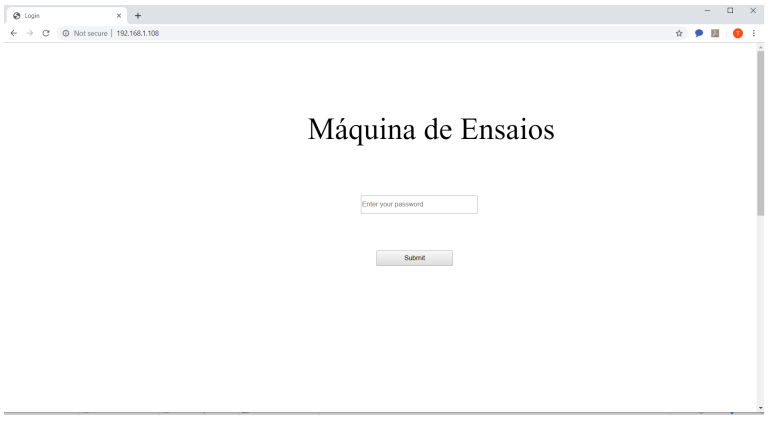

Figura 3.46: Menu Login da aplicação remota.

Depois de verificada a password a aplicação web verifica se a aplicação local está a correr e entra em modo de espera caso não esteja como é ilustrado na 3.47. Esta

verificação é feita recorrendo a um mecanismo de "file locking" que o Linux oferece. Quando a aplicação local inicia, faz um pedido ao sistema operativo para fazer locking a um ficheiro especifico e este locking só é libertado pelo sistema operativo quando o processo cessa. Ou seja, a aplicação web só tem de verificar a existência deste locking para se certificar que a aplicação local está a correr. Como esta verificação é feita recorrendo a uma system call do Linux que tem uma função wrapper no libc, o ficheiro php recorre à função "exec()" para executar uma programa auxiliar escrito em C cuja única função é verificar a existência deste "locking" e devolver o resultado ao script php que por sua vez o devolve à aplicação web.

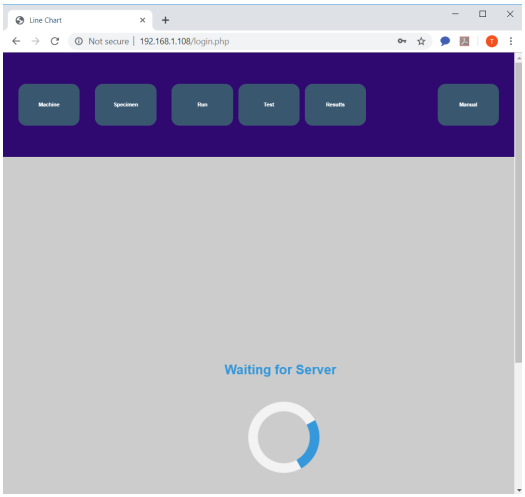

Figura 3.47: Aplicação remota em espera.

A comunicação entre a aplicação local e aplicação remota é feita recorrendo à memória partilhada. A memória partilhada é uma das formas de comunicação entre processos (IPC, Inter-Process Communications) do Linux que permitem a comunicação entre processos. A aplicação local escreve os valores a ser lidos pela aplicação remota nesta "shared memory". Do mesmo modo quando a aplicação remota necessita de enviar um comando, escreve na "shared memory". No caso do envio de comandos existe uma thread que faz constantemente pooling na shared memory para verificar se foi acionado algum comando. Tal como acontece na verificação do "file locking", em cada um dos casos acima referidos, o script php evoca um programa escrito em C que por sua vez lê ou escreve na "shared memory". O processo em que a aplicação remota lê valores da aplicação local é descrito na Figura 3.48 tem os seguintes passos:

1. A aplicação local lê o valor da posição e da força a cada 100ms
2. Estes valores são escritos na shared memory
3. A aplicação remota faz um pedido ajax para correr o script php values.php
4. O script php "updatemeasuresd.php" utiliza a função exec() para correr o processo "updatemeasures"
5. O processo lê os valores da "shared memory" e passa os valores ao script php

6. Os valores da posição e da força são enviados para a aplicação remota e o "Ajax Engine" atualiza a página html.

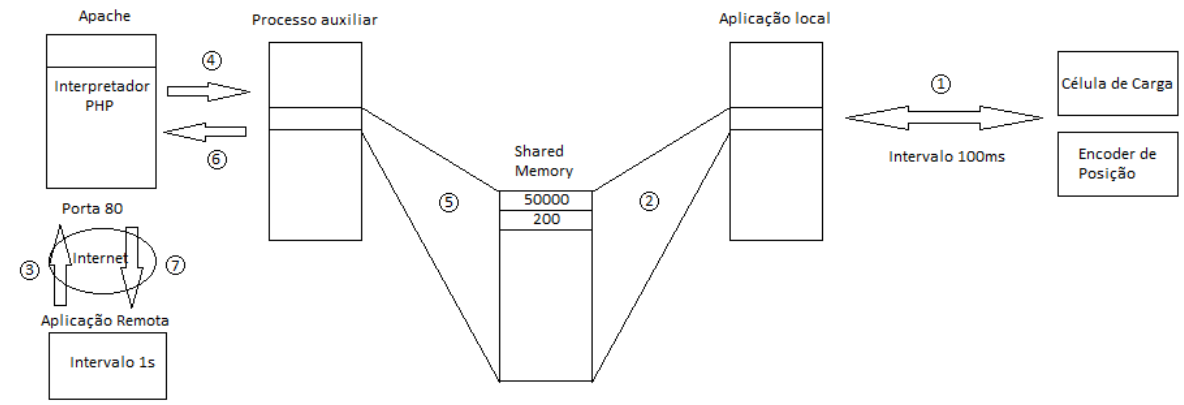

Figura 3.48: Exemplo da comunicação da aplicação remota com a aplicação local.

No caso de a aplicação remota querer enviar um comando para a aplicação local acontecem os seguintes passos e que estão descritos na Figura 3.49.

1. A aplicação remota faz um pedido ajax para correr o script php values.php
2. O script php "commands.php" utiliza a função exec() para correr o processo..
3. O processo "commands" escreve na shared memory os valores correspondentes ao comando. Neste caso o comando é para ser iniciado um ensaio de tracção com os respectivos campo
4. A thread 3 da aplicação local, responsável por fazer pooling na "shared memory" deteta o comando.
5. A partir daqui, a aplicação local inicia o ensaio como foi anteriormente explicado.

A thread 3 da aplicação local, ao ler o valor na "shared memory" zera os endereços de memória que leu. Assim tem a certeza que o comando é só uma vez executado e no caso do controlo manual, assegura que o utilizador na aplicação remota continua a pressionar o botão correspondente à ação desejada pois no próximo ciclo de leitura encontrará novamente o respetivo comando.

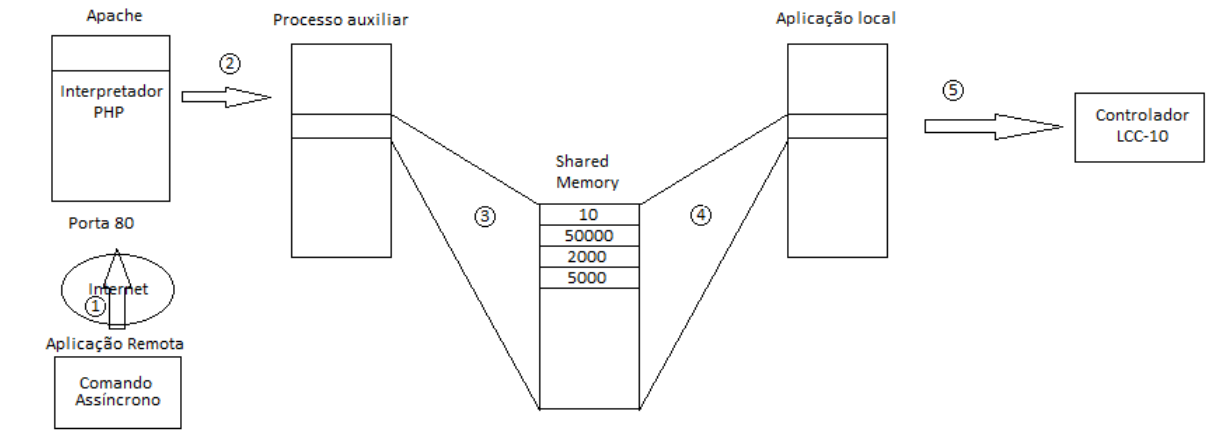

Figura 3.49: Exemplo da comunicação da aplicação remota com a aplicação local.

A troca de dados entre o servidor web e a aplicação remota é feita no formato JSON. No seguinte exemplo podemos ver a situação em que o utilizador clicou num determinado gráfico e cujos pontos são enviados neste formato.

```
{
    "x":["0","1","2","3","4","5","6","7","8"],
    "y":["15","20","19","10","25","33","35","38","39"]
}
```

3.6 Instalação do Software

- Instalar a última versão do sistema operativo Raspbian disponível em "https:///www.raspberrypi.org//d
- Na shell do Linux entrar na pasta "testing_machine". Tudo o que é necessário à instalação encontra-se nesta pasta.
- Dentro da pasta kplot fazer "sudo make" e posteriormente "sudo make install" para instalar esta biblioteca necessária à renderização dos gráficos de resultados.
- Instalar o servidor web Apache fazendo "sudo apt-get install apache2".
- Instalar o interpretador php fazendo "sudo apt-get install php".

- Instalar a base de dados MariaDB fazendo "sudo apt-get install mysql-server".
- Configurar a base de dados fazendo "sudo mysql -u root -p", "CREATE USER local@localhost IDENTIFIED BY 'maquina"' e posteriormente "GRANT ALL PRIVILEGES ON *.* TO local@localhost".
- fazer "./configdatabase" para correr o executável fornecido e assim configurar toda a base de dados.
- Dentro da pasta src fazer "sudo make install" para instalar a aplicação local e a aplicação web.
- A aplicação local é instalada na pasta "/usr/share/testing-machine" e a aplicação web é instalada na pasta "var/www/html".

3.7 Desenvolvimento das placas pcb

Neste projeto foram desenvolvidas duas placas pcb de forma a agrupar todos os componentes eletrónicos usados no mesmo. As placas foram desenvolvidas no programa gratuito Eagle. A primeira placa encaixa no Raspberry Pi enquanto a segunda encaixa na primeira. Em termos de dimensões, ambas as placas tem dimensões iguais às do Raspberry Pi.

O esquema elétrico da primeira placa está ilustrado na Figura 3.51 e o seu esquema físico na Figura 3.52. Esta placa contém o circuito integrado Max232n responsável pela comunicação RS232 e o decoder LS7633R responsável pela leitura da posição. O header fêmea desta placa encaixa no header macho da placa do Raspberry Pi. O header macho da primeira placa encaixa no header fêmea da segunda placa.

O esquema elétrico da segunda placa está ilustrado na Figura 3.53 e o seu esquema físico na Figura 3.54. Esta placa agrupa os conectores dos sinais vindos do encoder, dos sinais vindos da célula de carga e dos sinais da comunicação RS232. Também contém o módulo HX711 e o circuito de potência que permite a alimentação do controlador LCC-11 e do motor elétrico da máquina de ensaios. Quer o controlador, quer o motor elétrico são alimentados por uma tensão de 48V DC. Para este fim é usada uma fonte de alimentação com esta tensão. Para fazer a comutação desta tensão é usado um mosfet de potência.

O mosfet de potência usado neste projeto é o IRF540 e é acionado por um optoisolador fotovoltaico TLP591B ilustrado na Figura 3.50. Ao fazer-se passar uma dada corrente no elemento emissor, é gerada uma tensão entre os terminais 4 e 6 devido ao array de elementos fotovoltaicos. Assim, ligando o terminal 4 à source do mosfet de potência e o terminal 6 ao seu gate, é possível ligar e desligar o motor e o controlador LCC-11 consoante o estado de um pino do Raspberry Pi ligado ao terminal 1. O conjunto controlador e motor são representados no esquema elétrico da Figura 3.53 com a designação M1 e a fonte de alimentação é representada com a designação PS1. Como é possível ver neste esquema, o terminal positivo da fonte de tensão liga ao primeiro pino do conetor mais à esquerda enquanto o terminal negativo da fonte de tensão liga ao primeiro pino do conetor mais à direita. O controlador e o motor do atuador que estão ligados em paralelo, ligam ao segundo pino do conetor mais à esquerda e ao segundo pino do conetor mais à direita ficando assim em paralelo com o díodo de proteção também chamado de "flywheel".

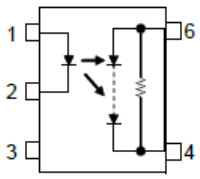

Figura 3.50: Opto-isolador fotovoltaico.

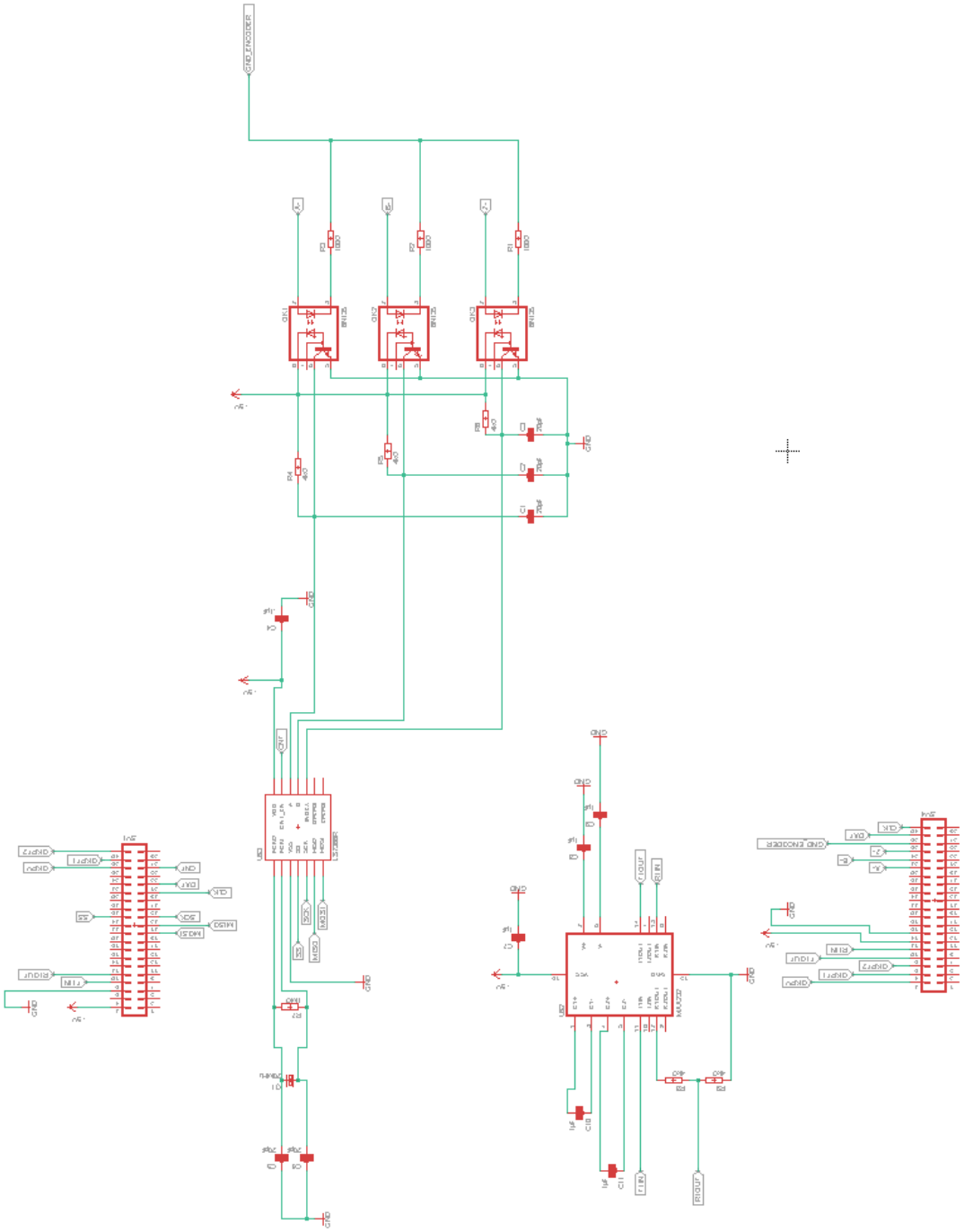

Figura 3.51: Esquema elétrico da placa 1 no Eagle.

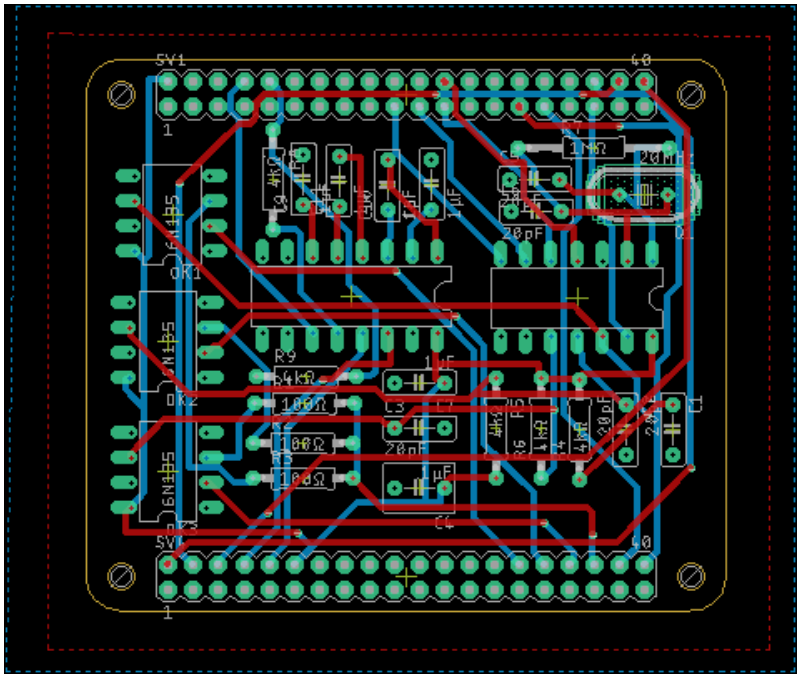

Figura 3.52: Layout da Placa 1 no Eagle.

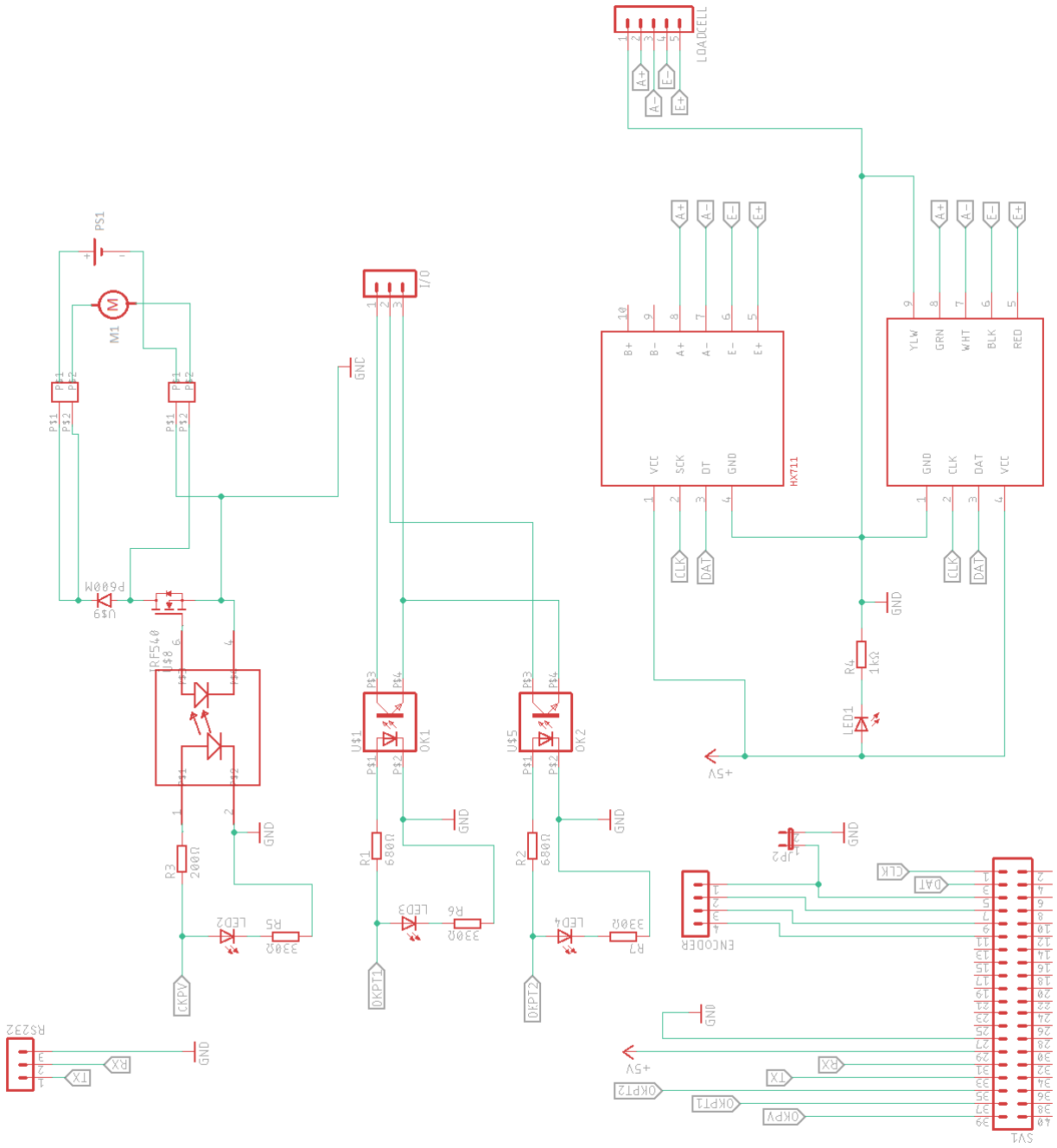

Figura 3.53: Esquema elétrico da placa 2 no Eagle.

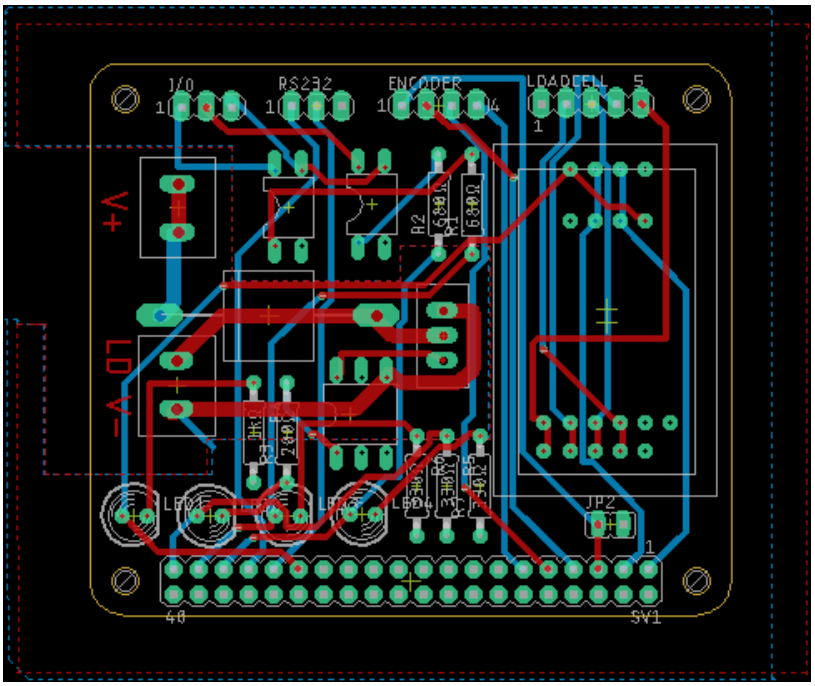

Figura 3.54: Layout da Placa 2 no eagle.

Capítulo 4

Conclusão

Com este projeto pode-se concluir que o computador Raspberry Pi foi uma boa solução encontrada para fazer todo o controlo da máquina de ensaios. Uma das grandes vantagens é ser económico, de tamanho reduzido e permitir a adição de placas pcb que o complementam. Para além disso todo o software usado é open source desde o sistema operativo à base de dados, assim como outro software necessário. O Raspberry Pi mostrou também ser uma boa solução para servir de Servidor Web correndo o Apache 2 e assim num só aparelho englobar todas as funcionalidades necessárias ao projeto.

O módulo HX711 com uma frequência máxima de 10hz foi suficiente para uma aquisição a cada 100ms, um valor bastante razoável. O decoder LS7633R mostrou ser uma solução eficaz para transformar a leitura do encoder em valores que o Raspberry Pi pudesse ler. Da mesma forma o circuito integrado MAX232N permitiu fazer a conversão dos níveis de tensão usados pelo Raspberry Pi para os níveis de tensão usados pelo controlador LCC-11 numa comunicação RS232.

Recorrendo ao Eagle foi possível desenhar as placas pcb que encaixam no Raspberry Pi de modo a torna a solução mais compacta. Com o mosfet de potência escolhido foi possível todo o controlo do LCC-11 e do motor elétrico.

Entretanto, no decorrer deste projeto foi lançado uma nova versão do Raspeberry Pi: a versão 4 B. Esta versão conta com um aumento significativo de ram de 1gb para 4gb, maior poder de processamento e uma conectividade melhorada nomeadamente o Wi-fi e a Ethernet.

Referências

[1] F. P. Beer, E. R. Johnston Jr., J. T. Dewolf, D. F. Mazurek, MECHANICS of MATERIALS.

[2] R. H. Bishop, The Mechatronics Handbook 2nd Edition.

[3] C. W. Silva, Sensors and Actuators Engineering System Instrumentation 2nd Edition.

[4] J. R. Davis, Tensile Testing Second Edition.

[5] M. S. Loveday, T. Gray, J. Aegerter, Testing of Metallic Materials: A Review

[6] G. Halfacree, Official Raspberry Pi Beginner's Guide

[7] J. F. Kurose, K. W. Ross, Computer Networking : A Top-Down Approach - 6th ed.

[8] M. Delamater, murach's JavaScript 2nd Edition.

[9] J. Murach, R. Harris, murach's PHP and MySQL 2nd Edition.

[10] S. Prata, C Primer Plus Sixth Edition.

[11] J. Masters, R. Blum, Professional Linux Programming.

[12] R. Love, Linux Kernel Development - Third Edition.

[13] A. Silberschatz, H. F. Korth, S. Sudarshan, Database System Concepts Sixth edition.

[14] mariadb.org/about/ 30-9-2019

[15] SMAC, LCC-10 Controller Specification and Features

[16] INGENIA-CAT, Embedded Motion Control Library - Product manual

[17] https://www.raspberrypi.org/ 7-10-2019